

**Rechnerstruktur und Ablaufgeschehen
in Rechenanlagen
Analysis und Simulation**

Der Technischen Fakultät
der Universität Erlangen-Nürnberg
zur Erlangung des Grades
DOKTOR - INGENIEUR

vorgelegt

von

Peter Paul Spies

Erlangen 1969

1969 № 386

**Rechnerstruktur und Ablaufgeschehen
in Rechenanlagen
Analysis und Simulation**

Der Technischen Fakultät
der Universität Erlangen-Nürnberg
zur Erlangung des Grades
DOKTOR - INGENIEUR

vorgelegt

von

Peter Paul Spies

Erlangen 1969



Als Dissertation genehmigt von der
Technischen Fakultät
der Universität Erlangen-Nürnberg

Tag der Einreichung: 30.5.1969

Tag der Promotion: 28.7.1969

Dekan: Prof. Dr. B. Ilschner

Berichterstatter: Prof. Dr. W. Händler

Prof. Dr. H. Billing

Inhalt

0. Einleitung	1
1. Ein Modell für Programmläufe	8
2. Rechenanlagen ohne Speicherhierarchie	30
3. Rechenanlagen mit Speicherhierarchie	68
4. Speicherstrategien	109
5. Ausblick: Adaptive Betriebsprogramme	121
Literatur	129



Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41546217_0020

0. Einleitung

Elektronische Digitalrechner sind heute unentbehrliche Hilfsmittel der Informationsverarbeitung. Der k l a s s i s c h e U n i v e r s a l r e c h n e r geht in seiner Konzeption auf Untersuchungen von Burks, Goldstine und von Neumann zurück; er besteht aus Speicher, Leitwerk, Rechenwerk, Eingabegerät und Ausgabegerät. Daten, die verarbeitet werden sollen, und Programme werden binär codiert im Speicher abgelegt. Ein- und Ausgabegerät dienen dem Informationsaustausch zwischen dem Speicher und der Außenwelt. Im Rechenwerk wird die eigentliche Verarbeitung der Daten durchgeführt. Das Leitwerk schließlich besitzt die Fähigkeit, die Gesamtheit dieser Aktivitäten zu steuern. Es entnimmt die Befehle eines auszuführenden Programmes dem Speicher und steuert ihre Ausführung in Zusammenarbeit mit den übrigen Werken bis schließlich alle in einem Programm festgelegten Verknüpfungen ausgeführt sind und die Endergebnisse im Speicher stehen oder ausgegeben sind.

Zur Charakterisierung der Eigenschaften m o d e r n e r R e c h e n a n l a g e n sind zwei Aspekte zu berücksichtigen: ihre gerätemäßige Ausstattung - hardware - und ihre Ausstattung mit Programmen - software genannt.

Zur h a r d w a r e : An die Stelle des Speichers im klassischen Universalrechner ist eine Hierarchie von Speichern unterschiedlicher Kapazität und mit unterschiedlicher Ausführungsdauer für Such-, Lese- und Schreiboperationen getreten. Eine typische Speicherhierarchie enthält einen Feritkernspeicher als Primärspeicher, eine Magnettrommel als Sekundärspeicher, Magnetplatten als Tertiärspeicher und Magnetkarten bzw. Magnetbänder als Quartär- und Archivspeicher. Für die Ein- und Ausgabe der Information steht ein

breites Spektrum von Geräten zur Verfügung; es umfaßt konventionelle Geräte wie Lochkartenleser und -stanzer ebenso wie Sichtgeräte (Kathodenstrahlröhren) mit Tastatur und Lichtstift. Schließlich erwies es sich, bedingt durch die verschieden langen Operationszeiten, als zweckmäßig, die Steuerung der internen Verknüpfungen und der Ein- bzw. Ausgabeoperationen von verschiedenen Leitwerken ausführen zu lassen und diesen die Fähigkeit zu geben, gleichzeitig (simultan) zu arbeiten. Eine moderne Rechenanlage wollen wir uns wie folgt strukturiert vorstellen:

Ein oder mehrere R e c h n e r k e r n e (Central Processing Unit) führen die interne Verarbeitung durch. Jeder Rechnerkern übernimmt die Funktionen des Leit- und Rechenwerkes eines klassischen Universalrechners, jedoch ohne die Fähigkeit, Ein- und Ausgabeoperationen auszuführen. Ein Rechnerkern entnimmt Programme und Daten ausschließlich dem Primärspeicher, in dem er auch Ergebnisse ablegt.

Ein oder mehrere E / A - S t e u e r w e r k e (Channel) führen den Informationsaustausch zwischen den Ein- und Ausgabegeräten und dem Primärspeicher und zwischen den Hintergrundspeichern (Sekundär-, Tertiärspeicher usw.) und dem Primärspeicher aus. Sie arbeiten dabei nach Programmen, die im Primärspeicher abgelegt sind. E/A - Steuerwerke werden von einem Rechnerkern zur Arbeit aktiviert.

Z u r s o f t w a r e : Die ersten Rechner wurden in Maschinensprache programmiert. Dies setzt beim Programmierer eine genaue Kenntnis der internen Struktur einer Rechenanlage voraus. Um einem großen Interessentenkreis den Zugang zur Rechenanlage zu ermöglichen, mußte die

Programmierung vereinfacht werden. Dies geschah durch folgende Maßnahmen: Es wurden Kunstsprachen geschaffen, die es dem Benutzer erlauben, Programme in einer problemnahen Form zu schreiben. Den Rechenanlagen wurde die Aufgabe übertragen, mit Hilfe von Übersetzungsprogrammen (Compiler) die so formulierten Programme in die Maschinsprache umzuwandeln. Zudem wurden für die verschiedensten Probleme sogenannte Unterprogramme erstellt, die, einmal programmiert, jedem Benutzer, der ein entsprechendes Problem zu bearbeiten hat, zur Verfügung gestellt werden, so daß er dieses ohne nennenswerten Programmieraufwand bearbeiten kann. Im folgenden werden diese Hilfen für den Benutzer als Arbeitsprogramme bezeichnet. Es ist heute üblich, daß die Hersteller von Digitalrechnern neben der hardware eine umfangreiche Programmbibliothek liefern. Andererseits ist es üblich, daß in einem Rechenzentrum mehrere Programmiersprachen und sonstige Arbeitsprogramme nebeneinander benutzt werden.

Die Arbeit einer so ausgestatteten Rechenanlage organisieren bedeutet somit, die Gesamtheit der verfügbaren hardware und die Gesamtheit der zur Erleichterung der Benutzung zur Verfügung stehenden Programme verwalten und den Benutzern je nach Bedarf zuteilen. Da manuelle Operationen immer mit einem beträchtlichen Zeitverlust verbunden und in hohem Maße mit Fehlern behaftet sind, ist es üblich die Organisation der Arbeit einer Rechenanlage von dieser selbst - abgesehen von den unumgänglichen manuellen Tätigkeiten - ausführen zu lassen, wobei sie von einem Betriebsprogramm gesteuert wird.

Es ergibt sich folgendes Gesamtbild:

Eine Rechenanlage besteht aus einer Menge von
G e r ä t e n (Rechnerkerne, E/A - Steuerwerke,
Speicherhierarchie, Ein- und Ausgabegeräte) und

einem Systemprogramm, das in den Speichern abgelegt ist. Das Systemprogramm besteht aus dem Betriebsprogramm und den Arbeitsprogrammen (Generatoren, Übersetzer, Programmtestprogramme, Maschinentestprogramme usw.). Geräte und Arbeitsprogramme werden als Betriebsmittel bezeichnet. Das Betriebsprogramm führt die zeitliche und räumliche Koordinierung der Arbeit der Rechananlage durch. Benutzer fordern vom Betriebsprogramm Betriebsmittel an und dieses teilt sie ihnen unter Berücksichtigung aller anstehenden Anforderungen zu.

Die Benutzung einer Rechananlage ist in vielen Fällen so organisiert, daß Programme am Rechenzentrum abgegeben und die Ergebnisse nach einer mehr oder weniger langen Wartezeit abgeholt werden. Der Benutzer hat keine Möglichkeit der unmittelbaren Einflußnahme während der Ausführung seiner Programme. Diese Art wird aber wohl in Zukunft mehr und mehr in den Hintergrund treten gegenüber Systemen, in denen die Rechananlage und ihre Umwelt zu einer geschlossenen informationsverarbeitenden Einheit zusammengefaßt sind (on - line Systeme), in denen wechselseitig Umwelt und Rechananlage Aufträge an den Partner geben und Ergebnisse von diesem entgegennehmen. Ansätze für die Realisierung dieser Konzeption wurden in Teilnehmer - Rechensystemen und in der Prozeßsteuerung realisiert. Teilnehmer - Rechensysteme (time - sharing Systeme) ermöglichen es einer größeren Anzahl von Benutzern, den Ablauf ihrer Programme von Außenstationen aus (meist Fernschreiber oder Sichtgeräte) unmittelbar zu überwachen und zu steuern. In der Prozeßsteuerung ist die Rechananlage so mit den Elementen eines physikalischen bzw. chemischen Prozesses verbunden, daß sie den Gesamtablauf kontrollieren und steuern kann.

Berücksichtigt man die Komplexität der Ablaufstrukturen, die in Systemen der beschriebenen Art zu steuern sind und die Mannigfaltigkeit der Anforderungen, die an eine Rechenanlage gestellt werden, so erscheint es unerlässlich, die heute vorhandenen Rechenanlagen und die Organisation ihrer Arbeit kritisch zu analysieren. Entspricht die vom klassischen Universalrechner, bei dessen Planung die Absicht im Vordergrund stand, die mechanische Tischrechenmaschine zu automatisieren, abgeleitete Struktur der Rechenanlagen den heutigen Anforderungen? Wie müsste eine Rechenanlage arbeiten, die den hohen Anforderungen an die Verfügbarkeit genügt, die in der Prozeßsteuerung gestellt werden müssen?

Die vorliegende Arbeit zeigt einige Methoden, die bei der Behandlung dieser Probleme angewandt werden können. Im Vordergrund des Interesses steht dabei die Frage, wie ein Betriebsprogramm die Tätigkeit einer Rechenanlage organisieren kann, wenn verschiedenartige äußere Anforderungen erfüllt und gleichzeitig die Elemente der Anlage wirtschaftlich ausgenutzt werden sollen. Die organisatorischen Maßnahmen eines Betriebsprogrammes werden dabei losgelöst von ihrer Implementierung betrachtet. Bei der Behandlung dieser Fragen ergeben sich gleichzeitig Aussagen über die Zweckmäßigkeit der Struktur von Rechenanlagen. Voraussetzung für die Untersuchungen sind geeignete Modellvorstellungen für Programme und für die Rechenanlagen, welche diese Programme ausführen sollen.

In K a p i t e l 1 wird ein wahrscheinlichkeitstheoretisches Modell für die Beschreibung von Programmabläufen angegeben, das für die Behandlung der Probleme, die hier im Vordergrund stehen, geeignet ist und bei einer Erweiterung der Problemstellung mit entsprechenden Modifikationen ebenfalls benutzt werden kann. Programme werden

beschrieben als "gewichtete Wege" durch eine virtuelle Rechenanlage. Die Eingabe eines Programmes entspricht der Forderung, Betriebsmittel bereitzustellen, damit ein Weg aus der Menge der möglichen Wege "durchlaufen" werden kann.

In den Kapiteln 2 und 3 werden Modelle für Rechenanlagen angegeben und untersucht. Die Strukturen realer Rechenanlagen sind so vielfältig, daß es zweckmäßig erscheint, mehrere Betrachtungsebenen zuzulassen und die Probleme der Ablaufplanung für diese getrennt zu behandeln. Jede Betrachtungsebene hebt gewisse Details der Rechenanlage als wesentlich hervor, während sie andere vernachlässigt. Dem "Auflösungsvermögen" dieser Modelle entsprechend werden auch die Programmabläufe beschrieben.

Die Modelle, welche im K a p i t e l 2 behandelt werden, berücksichtigen ausschließlich Rechnerkerne und E/A - Steuerwerke. Für diese lassen sich die Probleme der Ablaufsteuerung weitgehend mit den Mitteln der Warteschlangentheorie lösen.

In K a p i t e l 3 wird neben Rechnerkern und E/A - Steuerwerk eine dreistufige Speicherhierarchie berücksichtigt. Der Speicherbedarf eines Programmes wird dabei als unteilbare Einheit angesehen. An diesem Modell werden Untersuchungen durchgeführt, die zur Klärung der Frage dienen, welche Belastung Umspeicherprozesse zwischen den Elementen der Speicherhierarchie in Abhängigkeit von ablaufsteuernden Strategien mit sich bringen. Dieses Modell wird durch stochastische Simulation analysiert.

Betrachtet man den Speicherbedarf eines Programmes nicht mehr als unteilbare Einheit, so werden für die Ablaufsteuerung neue Möglichkeiten dadurch erschlossen, daß die Ausführung eines Programmes auch dann möglich ist, wenn im Primärspeicher nur ein Teil des Programmes und der zugehörigen Daten abgelegt ist. Für das Laden der einzelnen Programmteile sind dann verschiedene Vorgehensweisen möglich. In K a p i t e l 4 wird untersucht, welcher Zusammenhang zwischen der Anzahl der notwendigen Umspeicherprozesse und dem zugewiesenen Primärspeicherraum besteht.

In K a p i t e l 5 wird ausgeführt, wie die Organisation der Tätigkeit einer Rechenanlage unter heterogenen Anforderungen ihrer Umwelt durch ein adaptives Betriebsprogramm gesteuert werden könnte. Hierbei handelt es sich um eine Konzeption, die nur skizziert werden kann. Ihre detaillierte Ausführung wird erst möglich sein, wenn ein umfassender Überblick über den Wert ablaufsteuernder Strategien erarbeitet ist.

Bei der Durchführung der vorliegenden Untersuchungen, insbesondere in den Kapiteln 3 und 4, zeigte es sich, daß auch in der Literatur eine fast vollständige Unkenntnis der Parameter von Programmen herrscht, welche für die Ablaufsteuerung wesentlich sind. Für die Weiterführung entsprechender Analysen ist es dringende Voraussetzung, Aussagen über diese Parameter zu gewinnen.

1. Ein Modell für Programmläufe

1.0. Vorbemerkungen

Alle Untersuchungen zu der Frage, welchen Einfluß die Struktur einer Rechenanlage auf deren Fähigkeiten bei der Ausführung von Programmen hat, oder zu der Frage, wie ein Betriebsprogramm die Tätigkeiten einer Rechenanlage (mit vorgegebener Struktur) organisieren kann, wenn von den Benutzern besondere Anforderungen bzgl. der Fertigstellungstermine gestellt werden, setzen voraus, daß eine angemessene Beschreibung für Programmausführungen vorliegt. Diese Beschreibung ist zweckmäßig so zu wählen, daß sie alle Details vernachlässigt, welche keinen Einfluß haben auf die beabsichtigten Untersuchungen.

In diesem Kapitel wird eine derartige Beschreibung angegeben. Da wir ein Programm als eine Anforderung von Betriebsmitteln ansehen, also als Anforderung von Geräten und Arbeitsprogrammen, sind beide Betriebsmitteltypen in die Beschreibung einzubeziehen. Bei den Untersuchungen organisatorischer Maßnahmen ist von besonderem Interesse, welcher Aufwand für die Durchführung der Organisation zu leisten ist. Für unsere Untersuchungen sind folglich die Operationen bei der Ausführung eines Programmes, die eine Nutzleistung im Sinne eines Benutzers darstellen, zu unterscheiden von Operationen bei der Ausführung desselben Programmes, welche für die Organisation der Gesamtarbeit der Rechenanlage nötig sind.

Zur Erläuterung der Problemstellung und zur Klärung einiger Grundbegriffe skizzieren wir zunächst eine Programmausführung in einer realen Rechenanlage.

Dann wird an einem Beispiel der Einfluß aktueller Parameter auf die Ausführung eines Programmes aufgezeigt. Schließlich wird einer Klasse von realen Rechenanlagen eine virtuelle Rechenanlage als Idealisierung gegenübergestellt und die Menge der möglichen Ausführungsabläufe eines Programmes als Wahrscheinlichkeitsraum - hier Auftrags - Raum genannt - beschrieben.

1.1. Ablauf der Programmausführung in einer realen Rechenanlage

Wir betrachten eine Rechenanlage, die insbesondere über folgende Geräte verfügt: Rechnerkern, E/A - Steuerwerk, dreistufige Speicherhierarchie, Lochkartenleser und Zeilendrucker.

Ein Benutzer lege ein Programm Π_1 , geschrieben in ALGOL, als Lochkartenstapel vor. Es soll - dies wird auf sogenannten Steuerkarten festgelegt - übersetzt und - es sei frei von syntaktischen Fehlern - anschließend ausgeführt werden; Ergebnisse seien über den Zeilendrucker auszugeben. Wir nehmen an, daß sich außer diesem noch andere Programme gleichzeitig zur Ausführung in der Anlage befinden können. Der Lochkartenstapel wird zunächst von einem E/A - Steuerwerk in den Tertiärspeicher gelesen; dieser Vorgang wird von einem speziellen Arbeitsprogramm gesteuert. Die Information wird in einen Pufferbereich des Primärspeichers gelesen und, wenn dieser Bereich gefüllt ist, umgespeichert. Die Steuerkarten, die mit den Karten des Programmes gelesen wurden, legen fest, daß die eingegebene Information vom ALGOL - Compiler übersetzt werden soll. Ist - evtl. nach einer Wartezeit - der Rechnerkern ver-

füßbar und im Primärspeicher genügend freier Raum, so werden der Compiler und das Programm π_1 in den Primärspeicher gelesen; die Übersetzung wird ausgeführt. Zur Vereinfachung der Beschreibung sei angenommen, der Compiler erzeuge unmittelbar eine ausführfähige Fassung von π_1 in Maschinensprache und nach Abschluß der Übersetzung beginne der Rechnerkern unverzüglich mit der Ausführung von π_1 . Vor dem Abschluß dieser Tätigkeit werde ein Programm π_2 eingelesen, von dem festgelegt sei, daß seine Ausführung Vorrang vor allen anderen Programmen hat. Das bedeutet, daß die Ausführung von π_1 vom Betriebsprogramm unterbrochen wird und der Rechnerkern zunächst π_2 bearbeiten muß. Dieser Regiewechsel kann folgende Konsequenzen haben:

1. Fall: Im Primärspeicher sollen beide Programme gleichzeitig gespeichert werden können; zum Regiewechsel ist die aus der Arbeit an π_1 resultierende Information aus den Registern des Rechnerkerns in den Primärspeicher zu übertragen (insbesondere der Inhalt des Befehlszählers) und der Befehlszähler mit der Adresse des ersten auszuführenden Befehls von π_2 zu laden. Nach Abschluß von π_2 wird der Rechnerkern vom Betriebsprogramm wieder in den Zustand bzgl. der Ausführung von π_1 versetzt (durch Laden der Register mit der früher abgespeicherten Information), in dem er sich zur Zeit der Unterbrechung befand, und π_1 wird weiter bearbeitet. In diesem Fall sind die Operationen, die zur Durchführung des Regiewechsels nötig sind, in wenigen Mikrosekunden ausgeführt.
2. Fall: Im Primärspeicher könne jeweils nur eines der beiden Programme gespeichert werden. Dann muß, nachdem die Ausführung von π_1 , wie in Fall 1

beschrieben, unterbrochen ist, zunächst π_1 (einschließlich Daten) aus dem Primär- in den Sekundärspeicher übertragen werden; dann kann π_2 in den Primärspeicher geladen und vom Rechnerkern bearbeitet werden. Nach Abschluß von π_2 ist ein Umspeicherprozeß bzgl. π_1 in umgekehrter Richtung nötig, bevor π_1 wieder vom Rechnerkern bearbeitet werden kann. Jeder einzelne dieser Umspeicherprozesse dauert wegen der langen Zugriffszeiten zu den Hintergrundspeichern mehrere Millisekunden, so daß hier der Zeitaufwand für den Regiewechsel größenordnungsmäßig der tausendfache von Fall 1 ist.

Zur weiteren Ausführung des Programmes sei bemerkt, daß jede in π_1 auftretende Ein- oder Ausgabeoperation einen Regiewechsel zwischen Rechnerkern und E/A - Steuerwerk bewirkt. Für diese gilt folgendes: Kann der Rechnerkern unverzüglich mit der Ausführung eines weiteren im Primärspeicher stehenden Programmes beginnen, so ist der Zeitaufwand für diesen Regiewechsel dem in Fall 1 angegebenen vergleichbar; ist dies nicht der Fall, so muß der Rechnerkern warten, bis das E/A - Steuerwerk seine Operationen abgeschlossen hat. Da auszugebende Ergebnisse zunächst im Hintergrundspeicher gesammelt werden, hat diese Wartezeit ebenfalls die Größenordnung eines Umspeicherprozesses. Ist die Bearbeitung von π_1 im Rechnerkern abgeschlossen, so werden alle Ergebnisse - analog zur Programmeingabe - von einem E/A - Steuerwerk über den Zeilendrucker ausgegeben.

In dem skizzierten Ablauf sind als Nutzleistung im Sinne der Ausführung von π_1 anzusehen: Einlesen und Übersetzen des ALGOL - Programmes, Ausführung des

erzeugten Maschinenprogrammes und Ausgabe der Ergebnisse. Das Betriebsprogramm teilt π_1 die Betriebsmittel Kartenleser, E/A - Steuerwerk, Speicherraum, Rechnerkern und Zeilendrucker, sowie die Arbeitsprogramme für die Ein- und Ausgabe und die Übersetzung zu. Zusätzlich berücksichtigt das Betriebsprogramm als Koordinator der Gesamtarbeit der Anlage die Dringlichkeit des Programmes π_2 , was im Fall 2 Umspeicherprozesse nötig macht, die keinen Beitrag zur Berechnung der Ergebnisse von π_1 oder π_2 leisten.

Allgemein kann die Programmausführung in einer realen Rechenanlage wie folgt beschrieben werden: Das Betriebsprogramm kennt eine Steuersprache. Die Eingabe eines Steuerkommandos (unter Einbeziehung aktueller Parameter) bedeutet, daß ein Auftrag an die Rechenanlage vorliegt. Zur Ausführung jedes Auftrages sind Betriebsmittel erforderlich, und zwar Arbeitsprogramme und Geräte. Das Betriebsprogramm registriert die Gesamtheit der anstehenden Betriebsmittelanforderungen in Wartelisten. Es erfüllt die Anforderungen, indem es nach bestimmten Kriterien auswählend, Programmausführungen im Rechnerkern und in E/A - Steuerwerken initiiert. Der Abschluß der Ausführung dieser Programme wird dem Betriebsprogramm gemeldet; das Betriebsprogramm wird aktiviert. In gleicher Weise bewirkt jede Anforderung zusätzlicher Betriebsmittel durch ein gerade bearbeitetes Programm und das Auftreten bestimmter Signale (z. B. das Auftreten eines Fehlers oder eine Zeitmeldung der Uhr) eine Unterbrechung der Arbeit des gerade im Rechnerkern ausgeführten Programmes und eine Aktivierung des Betriebsprogrammes. Jedes Ereignis,

das eine Aktivierung des Betriebsprogrammes bewirkt, sei als Betriebsprogramm - Aufruf bezeichnet.

Eine Vorschrift, die für jeden möglichen Zustand der Wartelisten und jeden möglichen Betriebsprogramm - Aufruf eindeutig festlegt, welche Betriebsmittelanforderung als nächste zu erfüllen ist, heißt (ablaufsteuernde) Strategie eines Betriebsprogrammes.

Unter einem P r o g r a m m sei im folgenden immer eine endliche Folge von Anweisungen gemäß der Syntax der Maschinensprache verstanden. Es kann Arbeitsprogramme des Systemprogrammes und Benutzerprogramme enthalten. Ein A u f t r a g liege vor, wenn durch ein Steuerkommando ein Programm aufgerufen und mit aktuellen Parametern versorgt wird. Die Ausführung eines Programmes mit einer speziellen Menge von aktuellen Parametern sei P r o g r a m m l a u f genannt. Betriebsmittelanforderungen eines Auftrages beziehen sich ausschließlich auf Geräte.

1.2. Der Einfluß aktueller Parameter auf Programmläufe

Für ein vorgegebenes Programm sind, in Abhängigkeit von den aktuellen Parametern, verschiedene Programmläufe möglich. Zur Motivierung der in 1.3. angegebenen Charakterisierung der Menge der Programmläufe eines festen Programmes betrachten wir ein B e i s p i e l .

Ein Rechnerkern habe die in Tabelle 1 angegebene Befehlsliste. Im Primärspeicher sollen

<u>Bezeichnung</u>	<u>Wirkung</u>	<u>Dauer</u>
B, n	1. $\langle n \rangle \rightarrow AC$ 2. $\langle BZ \rangle + 1 \rightarrow BZ$ 3. $\langle\langle BZ \rangle\rangle \rightarrow BR$	1 1 1
T, n	1. $\langle AC \rangle \rightarrow n$, $\langle AC \rangle$ unverändert 2. $\langle BZ \rangle + 1 \rightarrow BZ$ 3. $\langle\langle BZ \rangle\rangle \rightarrow BR$	1 1 1
A, n	1. $\langle n \rangle \rightarrow HR$ 2. $\langle AC \rangle + \langle HR \rangle \rightarrow AC$ 3. $\langle BZ \rangle + 1 \rightarrow BZ$ 4. $\langle\langle BZ \rangle\rangle \rightarrow BR$	1 3 1 1
S, n	1. $\langle n \rangle \rightarrow HR$ 2. $\langle AC \rangle - \langle HR \rangle \rightarrow AC$ 3. $\langle BZ \rangle + 1 \rightarrow BZ$ 4. $\langle\langle BZ \rangle\rangle \rightarrow BR$	1 3 1 1
M, n	1. $\langle n \rangle \rightarrow HR$ 2. $\langle AC \rangle \cdot \langle HR \rangle \rightarrow AC$ 3. $\langle BZ \rangle + 1 \rightarrow BZ$ 4. $\langle\langle BZ \rangle\rangle \rightarrow BR$	1 7 1 1
V, n	1. Falls $\langle AC \rangle \geq 0$: $n \rightarrow BZ$ Falls $\langle AC \rangle < 0$: $\langle BZ \rangle + 1 \rightarrow BZ$ 2. $\langle\langle BZ \rangle\rangle \rightarrow BR$	1 1
H	Keine Veränderungen	1

Zeichenerklärung: AC Akkumulator
 BZ Befehlszähler
 BR Befehlsregister
 HR Hilfsregister
 n Adresse einer Speicherzelle
 $\langle a \rangle$ Inhalt eines Registers oder einer Speicherzelle mit dem Namen a
 $a \circ b \rightarrow c$ Das Ergebnis der Verknüpfung von a mit b nach der Vorschrift $\circ$ steht nach Abschluß der Operation in dem Register bzw. der Speicherzelle mit dem Namen c.

Tabelle 1: Beispiel für eine Befehlsliste

die Zahlen a , b und c stehen, und zwar a in der Zelle mit der Adresse n_1 , b in n_2 und c in n_3 . Aus diesen sei die Zahl x zu berechnen nach der Vorschrift $x := a+b$, falls $a+b \geq 0$ und $x := (a+b)^2 \cdot c$, falls $a+b < 0$. x soll in n_4 gespeichert werden. Folgendes Programm führt die Berechnung von x durch:

Befehls- adresse	Befehl	Wirkung
p	B, n_1	$a \rightarrow AC$
p+1	A, n_2	$a+b \rightarrow AC$
p+2	T, n_4	$a+b \rightarrow n_4$
p+3	V, p+7	Sprung nach p+7, falls $a+b \geq 0$
p+4	M, n_4	$(a+b)^2 \rightarrow AC$
p+5	M, n_3	$(a+b)^2 \cdot c \rightarrow AC$
p+6	T, n_4	$(a+b)^2 \cdot c \rightarrow n_4$
p+7	H	Halt

} Wird nur aus-
geführt, falls
 $a+b < 0$

Die formalen Parameter dieses Programmes sind a , b und c .

Der Ablauf der Ausführung - fremde Einflüsse seien ausgeschlossen - ist vollständig festgelegt durch das Programm, die Befehlsliste und die Zahlenwerte für a , b und c . Von den Details, welche durch diese Angaben festgelegt werden, sind diejenigen irrelevant, welche keinen Einfluß auf die Betriebsmittelanforderungen haben. Würde man z. B. den zweiten Programmbefehl A, n_2 ersetzen durch den Befehl S, n_2 , so wären die Betriebsmittelanforderungen des so entstehenden Programmes genau dieselben. Wir können also die Bedeutung dessen, was in den einzelnen

Phasen der Ausführung geschieht, bis zu einem gewissen Grad vernachlässigen und einen Programmlauf dadurch beschreiben, daß wir angeben, welche Elemente oder Elementeteile der Rechenanlage zu einem gewissen Zeitpunkt der Ausführung für diese aktiviert sind; so ergeben sich "Wege" durch die Anlagenelemente. Andererseits legen gerade die aktuellen Zahlenwerte von a und b fest, ob $x := a+b$ oder $x := (a+b)^2 \cdot c$ berechnet wird, d. h. sie bestimmen, welcher der beiden möglichen "Wege" bei einem Programmlauf ausgewählt wird. Um dieser Tatsache Rechnung zu tragen ohne auf die Zahlenwerte von a und b im einzelnen einzugehen, nehmen wir folgendes an: über die Zahlenwerte von a und b , mit denen das Programm durchgerechnet wird, seien Wahrscheinlichkeitsaussagen bekannt, etwa die Aussage, daß in einem Drittel aller Fälle $a+b \geq 0$ und in zwei Dritteln aller Fälle $a+b < 0$ seien.

Bild 1 zeigt die beiden möglichen "Wege" des Programmbeispiels und Tabelle 2 enthält die Beschreibung der zugehörigen Programmläufe. Dabei ist zu berücksichtigen, daß die in Tabelle 1 angegebene Befehlsliste zur Vereinfachung die Operationsdauer eines Befehls als konstant bezeichnet; es ist jedoch bekannt, daß diese i. a. in Abhängigkeit von den aktuellen Parametern variiert. Entsprechendes gilt für die Ein- und Ausgabeoperationen, wobei die Streubreite wesentlich größer ist.

1.3. Die virtuelle Rechenanlage

Wir betrachten jetzt eine Rechenanlage, die als

Idealisierung realer Anlagen anzusehen ist. Für diese werden Zustände definiert, die in Abschnitt 1.4. dem Modell für Programmläufe zugrunde gelegt werden.

Die virtuelle Rechenanlage besitze folgende Eigenschaften:

1. Ihre Betriebsmittel seien:
 - ein Rechnerkern,
 - ein E/A - Steuerwerk,
 - ein homogener Speicher mit einer Kapazität von N Wörtern,
 - G Ein- bzw. Ausgabegeräte, die numeriert seien."Homogen" sei ein Speicher genannt, für den die Dauer einer Lese- oder Schreiboperation unabhängig von der Adresse der Speicherzelle ist, die von der Operation betroffen wird, zudem sollen Rechnerkern und E/A - Steuerwerk uneingeschränkt zu allen Speicherzellen Zugriff haben.
2. In ihrem Speicher seien das Programm und die aktuellen Parameter höchstens eines Auftrags enthalten.
3. Jedes Programm spreche einen Speicherbereich mit den Adressen $1, 2, \dots, N_1 \leq N$ an.
4. Es können höchstens folgende Operationen gleichzeitig ausgeführt werden:
 - 4.1. der Rechnerkern steuert eine Lese- oder Schreiboperation bzgl. einer Speicherzelle;
 - 4.2. das E/A - Steuerwerk führt eine Lese- oder Schreiboperation bzgl. einer Speicherzelle aus und steuert gleichzeitig ein Ein- oder Ausgabegerät.

Es wird angenommen, daß beim Start der Befehlszähler p enthält und eine Befehlsholphase initiiert ist.

Zustand	Operation	Dauer	
(1,0,p,0)	«BZ» → BR	1	
(1,0,n ₄ ,0)	«n ₄ » → AC	1	
(1,0,0,0)	«BZ» +1 → BZ	1	
(1,0,p+1,0)	«BZ» → BR	1	
(1,0,n ₂ ,0)	«n ₂ » → HR	1	
(1,0,0,0)	«AC» + «HR» → AC	3	
(1,0,0,0)	«BZ» +1 → BZ	1	
(1,0,p+2,0)	«BZ» → BR	1	
(1,0,n ₄ ,0)	«AC» → n ₄	1	
(1,0,0,0)	«BZ» +1 → BZ	1	
(1,0,p+3,0)	«BZ» → BR	1	Zu diesem Zeitpunkt verzweigt sich der Weg
(1,0,0,0)	p+7 → BZ	1	Dieser Zweig wird durch-
(1,0,p+7,0)	«BZ» → BR	1	laufen, wenn a+b ≥ 0 gilt.
(0,0,0,0)			
(1,0,0,0)	«BZ» +1 → BZ	1	
(1,0,p+4,0)	«BZ» → BR	1	
(1,0,n ₄ ,0)	«n ₄ » → HR	1	
(1,0,0,0)	«AC» · «HR» → AC	7	
(1,0,0,0)	«BZ» +1 → BZ	1	
(1,0,p+5,0)	«BZ» → BR	1	Dieser Zweig wird durch-
(1,0,n ₃ ,0)	«n ₃ » → HR	1	laufen, wenn a+b < 0 gilt.
(1,0,0,0)	«AC» · «HR» → AC	7	
(1,0,0,0)	«BZ» +1 → BZ	1	
(1,0,p+6,0)	«BZ» → BR	1	
(1,0,n ₄ ,0)	«AC» → n ₄	1	
(1,0,0,0)	«BZ» +1 → BZ	1	
(1,0,p+7,0)	«BZ» → BR	1	
(0,0,0,0)			

Für die Erklärung der Zeichen wird auf Tabelle 1 verwiesen.

Tabelle 2: Ablauf der Programmausführung

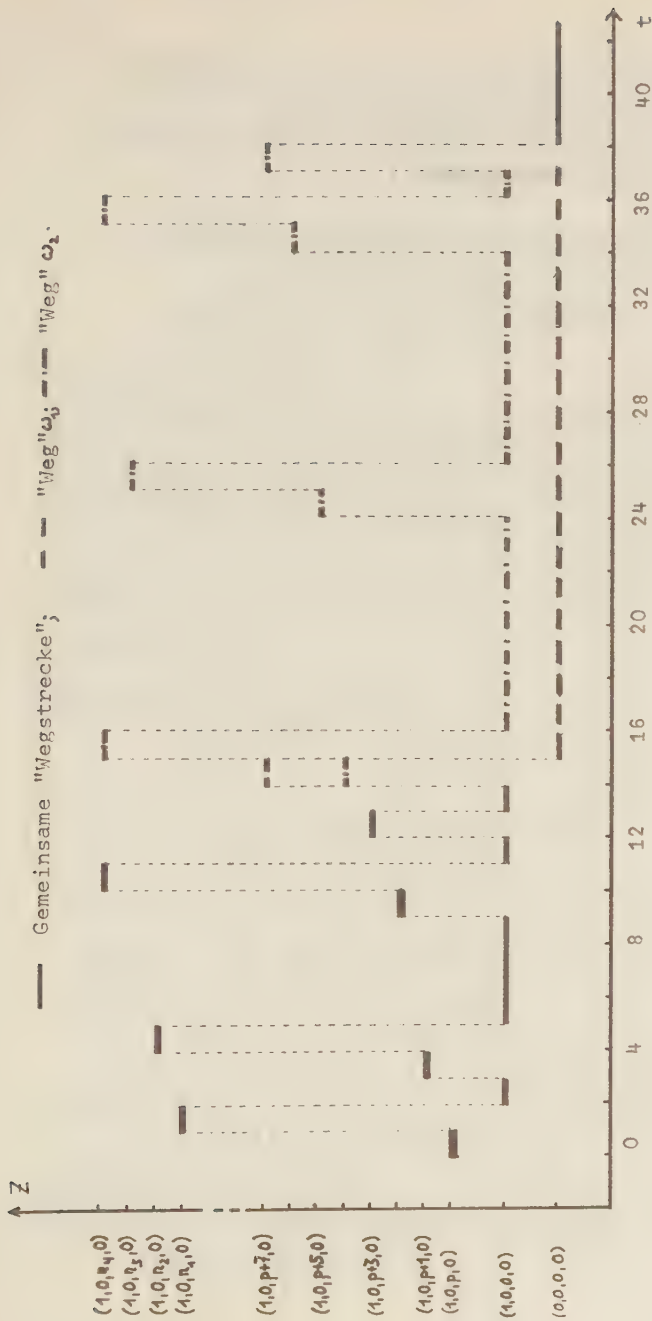


Bild 1: Beispiel für die "Wege" eines Programmes

Die Zustände, welche die virtuelle Rechenanlage annehmen kann, seien durch Quadrupel $z := (z_1, z_2, z_3, z_4)$ bezeichnet. Für die Komponenten sei folgendes festgelegt:

$z_1 \in \{0, 1\}$;

$z_1 = 0$ bedeute: der Rechnerkern führt die Operation "HALT" aus;

$z_1 = 1$ bedeute: der Rechnerkern führt eine Operation $\neq$ "HALT" aus.

$z_2 \in \{0, 1\}$;

$z_2 = 0$ bedeute: das E/A - Steuerwerk führe die Operation "HALT" aus;

$z_2 = 1$ bedeute: das E/A - Steuerwerk führe eine Operation $\neq$ "HALT" aus.

$z_3 \in \{0, 1, \dots, N\}$;

$z_3 = 0$ bedeute: im Speicher wird kein Lese- oder Schreibvorgang ausgeführt;

$z_3 = n$ mit $n \in \{1, \dots, N\}$ bedeute: in der Speicherzelle mit der Adresse n wird eine Lese- oder Schreiboperation ausgeführt;

$z_4 \in \{0, 1, \dots, G\}$;

$z_4 = 0$ bedeute: in keinem der Geräte wird eine Ein- oder Ausgabeoperation (einschließlich Suchvorgängen) ausgeführt;

$z_4 = n \in \{1, \dots, G\}$ bedeute: in dem Gerät mit der Nummer n wird eine Ein- oder Ausgabeoperation ausgeführt.

Mit diesen Festlegungen läßt sich jetzt die Zustandsmenge Z der virtuellen Anlage definieren.

Def. 1.1. Sei n eine beliebige natürliche Zahl, $n \in \mathbb{N}$, dann sei $Z_n := \{0, 1, \dots, n\}$.

Def. 1.2 Die Menge der Zustände, deren eine virtuelle Rechenanlage mit einer Speicherkapazität von N Wörtern und G Ein- und Ausgabegeräten fähig ist, sei Z mit

$$Z := \{(0,0,0,0)\} \cup \{z | z = (z_1, z_2, z_3, z_4) \in \mathbb{Z}_1 \times \mathbb{Z}_1 \times \mathbb{Z}_N \times \mathbb{Z}_G \\ \wedge z_1 \cdot z_2 = 0 \wedge z_1 \cdot z_4 = 0 \\ \wedge z_1 + z_2 = 1\}$$

1.4. Der Auftrags - Raum

Sei π ein fest vorgegebenes Programm, und A ein Auftrag bzgl. π . Wir nehmen an, die Ausführung von A erfolge in der virtuellen Rechenanlage, und wählen den Zeitpunkt des Starts als Koordinatenursprung. Für $t \geq 0$ sei mit $z_t(A)$ der Zustand der virtuellen Anlage zum Zeitpunkt t der Ausführung von A bezeichnet.

Def. 1.3. Sei $\mathbb{R}^+$ die Menge der nichtnegativen reellen Zahlen. Eine Abbildung $\omega: \mathbb{R}^+ \rightarrow Z$ heißt virtueller Programmlauf zum Auftrag A, wenn gilt:

1. Fall: Existiert ein $t_0 < \infty$ mit $z_{t_0}(A) = (0,0,0,0)$, so ist $\omega(t) = z_t(A)$ für alle $t \in [0, t_0]$ und $\omega(t) = (0,0,0,0)$ für alle $t > t_0$;
2. Fall: Existiert kein $t_0 < \infty$ mit $z_{t_0}(A) = (0,0,0,0)$, so ist $\omega(t) = z_t(A)$ für alle $t < \infty$ und $\omega(\infty) = (0,0,0,0)$.

Der virtuelle Programmlauf zeichnet also die Familie der Zustände nach, die im Verlauf der Ausführung eines Auftrages durch die virtuelle Rechenanlage angenommen werden.

Def. 1.4. Sei Ω_π die Menge der virtuellen Programmläufe von π für alle (endlich viele!) Aufträge bzgl. π , $\mathcal{P}(\Omega_\pi)$ die Potenzmenge von Ω_π und es existiere ein Wahrscheinlichkeitsmaß $P_\pi: \mathcal{P}(\Omega_\pi) \rightarrow [0,1]$ nach dem Aufträge bzgl. des Programmes π ausgewählt werden. Dann heißt das Tripel $(\Omega_\pi, \mathcal{P}(\Omega_\pi), P_\pi)$ Auftrags - Raum des Programmes π .

Die dritte Eigenschaft, die von einer virtuellen Rechenanlage gefordert ist, hat folgende Konsequenz: Existieren $\omega \in \Omega_\pi$ und $t \in \mathbb{R}^+$ mit $\omega(t) = (z_1, z_2, n, z_4)$, wobei $1 \leq n \leq N$ und z_1, z_2, z_4 beliebig sind, so existieren für jedes $m \in \{1, \dots, n-1\}$ ein $\omega' \in \Omega_\pi$ und ein $t' \in \mathbb{R}^+$ mit $\omega'(t') = (z_1', z_2', m, z_4')$ mit beliebigen z_1', z_2', z_4' . Diese Eigenschaft trägt der Tatsache Rechnung, daß Programmläufe invariant sind gegenüber Translationen in einem homogenen Speicher, sofern nur die Translation bzgl. aller Adressen, die in einem Programm auftreten, berücksichtigt ist.

Entsprechend trägt die Wahl des Startzeitpunktes eines Programmlaufes als Nullpunkt der Zeitachse der Tatsache Rechnung, daß Programmläufe invariant sind gegenüber Translationen der Zeitachse.

Beispiel:

Sei π das in Abschnitt 1.2. angegebene Programm. Wird eine virtuelle Rechenanlage gewählt, deren Rechnerkern die in Tabelle 1 angegebene Befehlsliste zugeordnet ist, so sind die virtuellen Programmläufe von π ω_1 und ω_2 mit:

$$\omega_1(t) := \begin{cases} (1,0,1,0) & \text{für } t \in [0,1) \\ (1,0,9,0) & \text{für } t \in [1,2) \\ (1,0,0,0) & \text{für } t \in [2,3) \\ (1,0,2,0) & \text{für } t \in [3,4) \\ (1,0,10,0) & \text{für } t \in [4,5) \\ (1,0,0,0) & \text{für } t \in [5,9) \\ (1,0,3,0) & \text{für } t \in [9,10) \\ (1,0,12,0) & \text{für } t \in [10,11) \\ (1,0,0,0) & \text{für } t \in [11,12) \\ (1,0,4,0) & \text{für } t \in [12,13) \\ (1,0,0,0) & \text{für } t \in [13,14) \\ (1,0,8,0) & \text{für } t \in [14,15) \\ (0,0,0,0) & \text{für } t \geq 15 \end{cases}$$

$$\omega_2(t) := \begin{cases} \omega_1(t) & \text{für } t \in [0,14) \\ (1,0,5,0) & \text{für } t \in [14,15) \\ (1,0,12,0) & \text{für } t \in [15,16) \\ (1,0,0,0) & \text{für } t \in [16,24) \\ (1,0,6,0) & \text{für } t \in [24,25) \\ (1,0,11,0) & \text{für } t \in [25,26) \\ (1,0,0,0) & \text{für } t \in [26,34) \\ (1,0,7,0) & \text{für } t \in [34,35) \\ (1,0,12,0) & \text{für } t \in [35,36) \\ (1,0,0,0) & \text{für } t \in [36,37) \\ (1,0,8,0) & \text{für } t \in [37,38) \\ (0,0,0,0) & \text{für } t \geq 38 \end{cases}$$

Somit ist $\Omega_\pi := \{\omega_1, \omega_2\}$ und P_π ist definiert durch
 $P(\emptyset) := 0$, $P(\{\omega_1\}) := \frac{1}{3}$, $P(\{\omega_2\}) := \frac{2}{3}$, $P(\Omega_\pi) := 1$.

Für jedes Programm, dessen Ablauf in den folgenden Kapiteln untersucht wird, nehmen wir an, daß der zugehörige Auftrags - Raum existiert und bekannt ist. Die in diesem angegebenen Eigenschaften legen die Programmläufe soweit fest, daß sie Untersuchungen zugrunde gelegt werden können, welche sich mit folgenden Fragen beschäftigen:

1. Fragen zur Struktur von Rechenanlagen:

Welchen Einfluß hat die Speicherhierarchie einer Rechenanlage auf die Ausführung von Aufträgen?
Welchen Einfluß hat die Anzahl funktionsgleicher Rechnerkerne auf die Ausführung von Aufträgen?
Welchen Einfluß hat die Anzahl funktionsgleicher E/A - Steuerwerke auf die Ausführung von Aufträgen?

2. Fragen zur Steuerung der Programmläufe:

(Die Struktur der Rechenanlage sei festgelegt.)
Welchen Einfluß hat die Reihenfolge, in der Aufträge ausgeführt werden, auf die Zeit, während der ein Auftrag unausgeführt in der Anlage verweilt?
Welchen Einfluß hat die Reihenfolge, in der Aufträge ausgeführt werden, auf die Auslastung der Elemente der Rechenanlage?

Der Auftrags - Raum ist, wie dies in Abschnitt 1.1. gefordert wurde, so gewählt, daß er Programmläufe unabhängig von der Anzahl der Rechnerkerne, der E/A-Steuerwerke und der Speicherhierarchie und ebenso frei von organisatorischen Maßnahmen, die keine Nutzleistung im Sinne eines Benutzers sind, der mit einem Programm gewisse Ergebnisse berechnen will, beschreibt. Die Annahme, daß die Operationen eines Programmes im wesentlichen seriell auszuführen sind, bedeutet heute keine wesentliche Beschränkung der Allgemeinheit, da Untersuchungen über die Parallelisierung von Algorithmen höchstens für Sonderfälle durchgeführt sind.

Andererseits gibt die Modellvorstellung von einer virtuellen Rechenanlage einen Hinweis für den, der

eine Beschreibung der Läufe eines vorgelegten Programmes angeben möchte. Er hat Programmläufe in einer realen Rechenanlage zu beobachten und gerade diejenigen Details eines Laufes unberücksichtigt zu lassen, die bei dem Gedankenexperiment des Programmlaufes in der virtuellen Rechenanlage nicht auftreten.

Die Details der Programmläufe, die wesentlich sind für die Analyse der Struktur von Rechenanlagenelementen, etwa der Struktur des Rechnerkerns oder der E/A - Steuerwerke, sind in der angegebenen Definition des Auftrags-Raumes vernachlässigt. Durch eine Erweiterung der Zustandsmenge der virtuellen Rechenanlage kann jedoch ein Auftrags - Raum für diese Zwecke analog definiert werden. Ist z. B. zu untersuchen, ob ein Rechnerkern mit Funktionseinheiten ausgestattet werden soll, die logische Operationen, Festkomma - und Gleitkomma - Operationen simultan ausführen können, so wäre der Zustand $z=(1,0,0,0)$ der virtuellen Anlage aufzuspalten in die vier Zustände

- $z^{(1)} := (1,0,0,0)$ mit der Bedeutung: der Rechnerkern führt eine logische Operation aus;
- $z^{(2)} := (2,0,0,0)$ mit der Bedeutung: der Rechnerkern führt eine Festkomma - Operation aus;
- $z^{(3)} := (3,0,0,0)$ mit der Bedeutung: der Rechnerkern führt eine Gleitkomma - Operation aus;
- $z^{(4)} := (4,0,0,0)$ mit der Bedeutung: der Rechnerkern führt eine Operation aus, die keinem der drei vorgenannten Typen angehört.

Ein Auftrags-Raum, der auf der Zustandsmenge definiert ist, die sich so ergibt, könnte dann entsprechenden Untersuchungen zugrunde gelegt werden. Hierauf wird jedoch im folgenden nicht näher eingegangen.

Statt eines einzelnen Programmes betrachten wir im folgenden i. a. endliche Mengen von Programmen. Dann sei wieder ein Wahrscheinlichkeitsraum $(\Pi, \mathcal{P}(\Pi), P')$ gegeben, wobei Π die Menge der Programme und P' ein Wahrscheinlichkeitsmaß $P' : \mathcal{P}(\Pi) \rightarrow [0,1]$ sind $(\Pi, \mathcal{P}(\Pi), P')$ wird auch Programm - Raum genannt).

Der Wahrscheinlichkeitsraum $(\Pi, \mathcal{P}(\Pi), P')$ und die Auftrags - Räume $(\Omega_\pi, \mathcal{P}(\Omega_\pi), P_\pi)$ für die Programme $\pi \in \Pi$ legen einen Auftrags - Raum $(\Omega, \mathcal{P}(\Omega), P)$ fest, für den gelten:

$$1. \quad \Omega := \bigcup_{\pi \in \Pi} \Omega_\pi$$

$$2.1. \quad \text{Für alle } \omega \in \Omega \text{ gilt: } P(\omega) := \sum_{\pi (\pi \in \Pi \wedge \omega \in \Omega_\pi)} P'(\pi) \cdot P_\pi(\omega)$$

2.2. Für jede nichtleere Untermenge Ω' von Ω gilt:

$$P(\Omega') := \sum_{\omega (\omega \in \Omega')} P(\omega)$$

In diesem Sinne ist es für das Folgende gleichgültig, ob ein gegebener Auftrags - Raum sich auf ein Programm oder auf eine Menge von Programmen bezieht.

Eine Unterscheidung von Aufträgen nach den Programmen, auf die sie sich beziehen, ist dann erforderlich, wenn mehrere Programmläufe gemeinsam dieselbe Kopie eines Programmes benutzen, was bei eingangsinvarianter Programmstruktur praktisch auftreten kann. In diesem Fall hat die Speicherverwaltung zu berücksichtigen, ob eine Kopie des Programmes bereits im Primärspeicher steht oder nicht. Derartige Zusammenhänge zwischen mehreren Aufträgen seien aus den folgenden Betrachtungen ausgeschlossen. Beziehen sich mehrere Aufträge auf dasselbe Programm oder auf dasselbe Unterprogramm, so soll jeder Programmlauf über eine Kopie davon verfügen, die nur von ihm allein durchlaufen wird. Entsprechendes gilt für aktuelle Parameter.

1.5. Informationsreduktion mittels meßbarer Abbildungen

Gegeben sei ein Auftrags - Raum $(\Omega, \mathcal{F}(\Omega), P)$. Für Untersuchungen über den Ablauf von Programmen sind u. U. nicht alle Details von Interesse, welche in den Elementen von Ω wiedergegeben werden. Es ist daher zweckmäßig, die Untersuchungen nicht auf die Elemente von Ω zu basieren, sondern Ω so in eine Menge Ω' abzubilden, daß Ω' genau die benötigten Details eines Programmlaufes enthält und auch nur diese. Da neben Ω das Wahrscheinlichkeitsmaß P von besonderem Interesse ist, wird in die Abbildung von Ω in Ω' auch P einbezogen, so daß als Abbildungsergebnis wieder ein Wahrscheinlichkeitsraum entsteht. Dies leistet gerade eine meßbare Abbildung.

Sei Ω eine Menge und $\mathcal{F}$ eine σ -Algebra in Ω . Dann wird das Paar $(\Omega, \mathcal{F})$ als M e ß r a u m bezeichnet.

Def. 1.5. Seien $(\Omega, \mathcal{F})$ und $(\Omega', \mathcal{F}')$ Meßräume und f sei eine Abbildung von Ω in Ω' , $f: \Omega \rightarrow \Omega'$, f heißt $(\mathcal{F}, \mathcal{F}')$ - m e ß b a r, wenn gilt:
 $f^{-1}(F') \in \mathcal{F}$ für jedes $F' \in \mathcal{F}'$. Dabei ist
 $f^{-1}(F') := \{\omega \mid \omega \in \Omega \wedge f(\omega) \in F'\}$.

Spezielle meßbare Abbildungen sind stochastische Variable.

Def. 1.6. Sei $\mathcal{B}$ die σ -Algebra der Borelschen Mengen in $\mathbb{R}$. Ist eine Abbildung $f: \Omega \rightarrow \mathbb{R}$ $(\mathcal{F}, \mathcal{B})$ -meßbar, wobei $\mathcal{F}$ eine σ -Algebra in Ω ist, so heißt f s t o c h a s t i s c h e V a r i a b l e.

Für meßbare Abbildungen gilt:

Satz 1.1. Sei $f: \Omega \rightarrow \Omega'$ eine $(\mathcal{F}, \mathcal{F}')$ - meßbare Abbildung bzgl. $(\Omega, \mathcal{F})$ und $(\Omega', \mathcal{F}')$. Dann wird für jedes Wahrscheinlichkeitsmaß $P: \mathcal{F} \rightarrow [0,1]$ durch $P'(F') := P(f^{-1}(F'))$ ein Wahrscheinlichkeitsmaß $P': \mathcal{F}' \rightarrow [0,1]$ definiert.

Der Beweis des Satzes ist z. B. in Bauer (1968) nachzulesen, wobei zu beachten ist, daß $P(\Omega)=1$ und $f^{-1}(\Omega')=\Omega$, also $P'(\Omega') = 1$ gelten.

Beispiel:

Betrachten wir den in Abschnitt 1.4. angegebenen Auftrags - Raum mit $\Omega_{\pi} = \{\omega_1, \omega_2\}$ und dem Wahrscheinlichkeitsmaß P nach der dort gegebenen Definition.

Von ω_1 und ω_2 interessiere uns ausschließlich die Zeit, die vergeht, bis ω_1 bzw. ω_2 erstmals den Zustand $(0,0,0,0)$ durchläuft.

Sei also: $X: \Omega \rightarrow \mathbb{R}$ definiert durch:

$$\begin{aligned} X(\omega_1) &:= t_1 \text{ mit } \omega_1(t_1) = (0,0,0,0) \text{ und} \\ &\quad \omega_1(t) \neq (0,0,0,0) \text{ für alle } t < t_1, \\ X(\omega_2) &:= t_2 \text{ mit } \omega_2(t_2) = (0,0,0,0) \text{ und} \\ &\quad \omega_2(t) \neq (0,0,0,0) \text{ für alle } t < t_2, \\ \text{also } X(\omega_1) &= 15 \text{ und } X(\omega_2) = 38. \end{aligned}$$

Hier ist X stochastische Variable. X ist die Ausführungsdauer von π und kann durch eine Verteilungsfunktion gekennzeichnet werden. Sind $F(x)$ die Verteilungsfunktion von X und P das Bildmaß von P_{π} bzgl. X , so ist $F(x) := P(X \leq x)$ für $x \in \mathbb{R}$, also

$$F(x) = \begin{cases} 0 & \text{für } x < 15 \\ \frac{1}{3} & \text{für } 15 \leq x < 38 \\ 1 & \text{für } 38 \leq x \end{cases}$$

Nachdem hier prinzipiell angegeben wurde, wie "Vergrößerungen" der charakterisierenden Eigenschaften von Programmläufen gewonnen werden können, werden wir uns in den folgenden Kapiteln mit speziellen stochastischen Veränderlichen, die aus einem vorgegebenen Auftrags - Raum abgeleitet werden, beschäftigen.

Nach der Definition des Auftrags - Raumes spaltet sich die Analyse von Rechnerstrukturen in zwei Zweige auf:

Ein Zweig untersucht, welche Auftrags - Räume für den Einsatz von Rechenanlagen in den verschiedenen Anwendungsgebieten repräsentativ sind.

Der andere Zweig nimmt einen Auftrags - Raum als gegeben an und untersucht die Struktur von Rechenanlagen und Reihenfolgeprobleme unter der Voraussetzung, daß die Rechenanlage Aufträge aus diesem Auftrags - Raum auszuführen hat. Mit diesem Zweig beschäftigen sich alle nachfolgenden Ausführungen.

2. Rechenanlagen ohne Speicherhierarchie

2.0. Vorbemerkungen

In diesem Kapitel beschäftigen wir uns mit der Organisation der Programmläufe und der Struktur von Rechenanlagen, die mit einem hinreichend großen Primärspeicher ausgestattet sind, in dem alle anstehenden Aufträge (Programme und Daten) abgelegt werden können.

In Abschnitt 2.1. nehmen wir an, daß der gesamte Programmlauf von einem universellen Rechnerkern gesteuert wird. Unter der Voraussetzung, daß der Zeitverlust für Regiewechsel vernachlässigbar ist, wird für endlich viele Aufträge, die zu verschiedenen Zeitpunkten eingegeben werden können, eine Strategie angegeben, welche die mittlere Verweilzeit jedes Auftrages minimiert. Für unendlich viele Aufträge wird das Problem der Ablaufsteuerung als Aufgabe der Warteschlangentheorie formuliert. Hier sind vielfältige Aussagen in der Literatur bekannt. Wir zitieren im wesentlichen einen Satz, der das Problem der Ablaufsteuerung für eine gewisse Klasse von Strategien löst.

In Abschnitt 2.2. wird angenommen, daß von jedem Programmlauf Teile im Rechnerkern und andere Teile in einem E/A - Steuerwerk ausgeführt werden. Für spezielle Verteilungsfunktionen der Ausführungsdauer eines Auftrages werden die Zustandswahrscheinlichkeiten berechnet, wobei Rechnerkern und E/A - Steuerwerk simultan arbeiten können (Zeitmultiplex - Betrieb bzgl. eines Auftrages und Simultanarbeit bzgl. mehrerer Aufträge). Diese Systemstruktur wird verglichen mit zwei anderen: ein System mit einem universellen

Rechnerkern und ein System mit zwei funktions-
äquivalenten universellen Rechnerkernen. Diese
Vergleichsergebnisse geben eine Orientierungshil-
fe, wenn geeignete Systemstrukturen für die Bear-
beitung gewisser Aufgaben ausgewählt werden.

2.1. Reihenfolgeprobleme im klassischen Universalrechner

Wir betrachten eine reale Rechanlage, die über
einen u n i v e r s e l l e n R e c h n e r -
k e r n verfügt, der die Operationen der internen
Verarbeitung und die Ein- und Ausgabeoperationen
steuert, also die Eigenschaften des Leit- und Re-
chenwerkes eines klassischen Universalrechners be-
sitzt. Zudem verfüge die Anlage über einen Primär-
speicher, dessen Kapazität ausreiche, alle anstehen-
den Aufträge mit ihren Programmen und Daten aufzu-
nehmen.

Sei $(\Omega, \mathcal{P}(\Omega), P)$ ein Auftrags - Raum. Unter den ange-
gebenen Voraussetzungen ist die Unterscheidung
zwischen Zuständen der virtuellen Rechanlage mit
 $z_1 = 1$ und mit $z_2 = 1$ irrelevant. Ebenso ist die
Größe des von einem Auftrag belegten Speicherraumes
ohne Einfluß auf den Programmlauf. Da alle Operationen
von dem universellen Rechnerkern gesteuert werden,
sind auch die angesprochenen Ein- und Ausgabegeräte
jeweils verfügbar. Zusätzlich sei folgendes angenom-
men:

Regiewechsel bei der Ausführung von Programmläufen
seien zu jedem Zeitpunkt und ohne Zeitverlust mög-
lich; nach Abschluß eines Programmlaufes könne ohne
Zeitverlust mit einer weiteren Ausführung begonnen
und ebenso könne ein Programmlauf ohne Zeitverlust

zu Gunsten eines anderen Auftrages unterbrochen werden.

Unter diesen Voraussetzungen sind für die Ablaufplanung von jedem Auftrag der Zeitpunkt seiner Eingabe und die Dauer des zugehörigen Programmlaufes von Bedeutung. Die Eingabezeitpunkte können explizit angegeben werden (Abschnitt 2.1.1.) oder durch die Länge der Zeitintervalle zwischen zwei aufeinanderfolgenden Eingaben (2.1.2.) gekennzeichnet werden.

Sei $E: \Omega \rightarrow \mathbb{R}$ die Abbildung, die für alle $\omega \in \Omega$ definiert ist durch

$$E(\omega) := t_0 \text{ mit } \omega(t_0) = (0, 0, 0, 0) \text{ und} \\ \omega(t) \neq (0, 0, 0, 0) \text{ für alle } t \in [0, t_0).$$

E ist meßbar bzgl. $\mathcal{F}(\Omega)$ und der λ -Algebra der Borelmengen von $\mathbb{R}$, also stochastische Variable. E ist die virtuelle Ausführungsdauer der Aufträge von $(\Omega, \mathcal{F}(\Omega), P)$ und genüge der Verteilungsfunktion $f(x)$.

2.1.1. Eine endliche Anzahl von Aufträgen

Den Ausführungen dieses Abschnittes liegen folgende Annahmen zugrunde:

1. Es werden jeweils endlich viele Aufträge betrachtet, und zwar $n \in \mathbb{N}$
2. Von jedem Auftrag sei spätestens zum Zeitpunkt der Eingabe die Dauer des zugehörigen Programmlaufes exakt bekannt.
3. Die Länge jedes Zeitintervalls, das im folgenden auftritt ist ganzzahliges Vielfaches einer positiven Zahl τ (z. B. die kürzeste Schaltzeit in der Rechenanlage).

Die n Aufträge seien numeriert; die A u s -
f ü h r u n g s d a u e r sei in $\mathcal{A} := (e_1, \dots, e_n)$,
die A n k u n f t s z e i t p u n k t e in
 $\mathcal{A} := (t_1, \dots, t_n)$ angegeben.

Def. 2.1. Eine Vorschrift, die für jeden Zeit-
punkt eindeutig festlegt, welcher der
anstehenden Aufträge von der Bedienungs-
station ausgeführt werden soll, heiße ab-
laufsteuernde S t r a t e g i e .

Sei S eine ablaufsteuernde Strategie. S legt bei ge-
gebenen $\mathcal{A}$ und $\mathcal{A}$ für jeden Auftrag den Zeitpunkt fest,
zu dem der zugehörige Programmlauf abgeschlossen ist.
Diese seien F e r t i g s t e l l u n g s z e i t -
p u n k t e genannt und mit $\tau := \tau(S, \mathcal{A}) := (c_1, \dots, c_n)$
bezeichnet.

Def. 2.2. Bei gegebenen $S, \mathcal{A}$ und $\mathcal{A}$ ist
 $w_i(S, \mathcal{A}) := c_i - e_i - t_i$ die W a r t e -
z e i t des i -ten Auftrages für
 $i \in \{1, \dots, n\}$.

Def. 2.3. Bei gegebenen $S, \mathcal{A}$ und $\mathcal{A}$ ist
 $v_i(S, \mathcal{A}) := c_i - t_i$ die V e r w e i l -
z e i t des i -ten Auftrages für
 $i \in \{1, \dots, n\}$.

Wenn Strategien aus einer Menge Σ betrachtet werden,
so ist es häufig wünschenswert, diese in ihrer
Wirkung auf die Ausführung von Aufträgen miteinan-
der zu vergleichen. Hierzu werden die Strategien be-
wertet.

Def. 2.4. Eine Vorschrift ψ , die bei gegebenen n und t jedem S aus einer Menge Σ von Strategien eindeutig eine nichtnegative reelle Zahl zuordnet, heißt Bewertung von Σ bzgl. n und t . $\psi(S, n, t)$ heißt Wert von S bei n und t .

Beispiel:

Bei gegebenem n und t ordne $\bar{v}$ jeder Strategie S die mittlere Verweilzeit eines Auftrages bei der Ausführung gemäß S zu, also:

$$\begin{aligned}\bar{v}(S, n, t) &:= \frac{1}{n} \sum_{i=1}^n v_i(S, n, t) \\ &= \frac{1}{n} \sum_{i=1}^n (c_i(S, n, t) - t_i)\end{aligned}$$

Def. 2.5. Eine Bewertung ψ von Σ bzgl. n und t heißt regulär, wenn aus

$$\begin{aligned}\psi(S, n, t) &> \psi(S', n, t) \text{ folgt} \\ c_i(S, n, t) &> c_i'(S', n, t) \text{ für wenigstens} \\ &\text{ein } i \in \{1, \dots, n\}.\end{aligned}$$

Beispiel:

Sei ψ die Bewertung, die jeder Strategie $S \in \Sigma$ bzgl. n und t die mit $u := (u_1, \dots, u_n)$ und $u_i > 0$ für $i \in \{1, \dots, n\}$ gewichtete mittlere Verweilzeit zuordnet, also:

$$\begin{aligned}\psi(S, n, t) &:= \frac{1}{n} \sum_{i=1}^n u_i \cdot v_i(S, n, t) \\ &= \frac{1}{n} \sum_{i=1}^n u_i \cdot (c_i(S, n, t) - t_i)\end{aligned}$$

$$\psi(S, n, t) > \psi(S', n, t) \Leftrightarrow \sum_{i=1}^n u_i \cdot (c_i(S, n, t) - t_i) > \sum_{i=1}^n u_i \cdot (c_i'(S', n, t) - t_i) \Leftrightarrow$$

$$\sum_{i=1}^n u_i \cdot (c_i - c_i') > 0$$

Das ist nur möglich, wenn wenigstens einer der Summanden > 0 ; also ist ψ regulär.

Def. 2.6. Sei Σ eine Menge von Strategien und ψ eine Bewertung von Σ bzgl. n und t . Eine Strategie $S \in \Sigma$ heiÙe optimal bzgl. ψ , wenn $\psi(S, n, t) \leq \psi(S', n, t)$ gilt für alle $S' \in \Sigma$.

Def. 2.7. Sei S eine ablaufsteuernde Strategie bzgl. n und t . Ein Zeitintervall $[t', t'')$ mit $t' < t''$ heiÙt Leerstelle bzgl. S, n und t , wenn

1. S festlegt, daÙ die Bedienstationsstation während $[t', t'')$ keinen Auftrag ausführt und
2. wenigstens ein $i \in \{1, \dots, n\}$ mit $c_i(S, n, t) > t''$ und $t_i < t'$ existiert.

Im folgenden sei $T := \bigcup_{i=1}^n \{t_i\}$. Ohne Beschränkung der Allgemeinheit wird $\min T = 0$ angenommen.

Im allgemeinen erlauben Strategien Leerstellen und Ausführungsunterbrechungen. Speziell für Strategien, die optimal bzgl. regulärer Bewertungen sind, gelten folgende Aussagen.

Satz 2.1. Sei Σ die Menge aller Strategien und ψ eine reguläre Bewertung von Σ bzgl. n und t . Dann existiert zu jeder Strategie S' mit Leerstellen eine Strategie S ohne Leerstelle mit $\psi(S, n, t) \leq \psi(S', n, t)$.

Beweis: Sei $S' \in \Sigma$ und der von S' erzeugte Ablaufplan enthalte genau eine Leerstelle $[t', t'']$.

Sei S folgende Strategie:

1. für $[0, t']$ erzeuge S denselben Ablaufplan wie S' ;
2. für $t \geq t'$ erzeuge S einen Ablaufplan, der aus dem von S' erzeugten dadurch hervorgeht, daß die Ausführung jedes Auftrages mit c_i (S', n, t) $> t'$
 - 2.1. um $t'' - t'$ vorverlegt wird, falls der Ankunftsstermin dies zuläßt, bzw.
 - 2.2. zum Ankunftsstermin begonnen wird, falls die Vorverlegung um $t'' - t'$ nicht möglich ist.

Nach Def. 2.7. existiert wenigstens ein Auftrag, dessen Ausführung vorverlegt werden kann.

Nach Konstruktion enthält S keine Leerstelle. Ebenso folgt c_i (S, n, t) $\leq c_i$ (S', n, t) für alle $i \in \{1, \dots, n\}$.

Nach Def. 2.5. folgt also $\psi(S, n, t) \leq \psi(S', n, t)$.

Ist $S' \in \Sigma$ eine Strategie mit mehreren Leerstellen, so kann jede davon nach dem angegebenen Konstruktionsverfahren beseitigt werden; bei keinem derartigen Schritt wächst der Wert bzgl. ψ .

Folgerung: Aus Satz 2.1. folgt, daß es genügt, die Strategien ohne Leerstellen zu berücksichtigen, wenn eine bzgl. einer regulären Bewertung optimale Strategie gesucht ist.

Satz 2.2. Sei Σ die Menge aller Strategien und ψ eine reguläre Bewertung von Σ bzgl. n und t . Dann existiert zu jeder Strategie S' , die Ausführungsunterbrechungen außerhalb der Elemente von T zuläßt, eine Strategie S , die Ausführungsunterbrechungen höchstens in Zeitpunkten $t \in T$ zuläßt mit $\psi(S, n, t) \leq \psi(S', n, t)$

Beweis: Sei $S' \in \Sigma$ und S' definiere einen Ablaufplan, der genau eine Ausführungsunterbrechung außerhalb T festlege, und zwar zum Zeitpunkt $t_0 \notin T$.

Sei $t' < t_0 < t''$ und S' lege folgendes fest: Im Zeitintervall $[t', t_0)$ wird der Auftrag $i \in \{1, \dots, n\}$ bearbeitet; im Zeitintervall $[t_0, t'')$ wird der Auftrag $j \in \{1, \dots, n\}$ mit $i \neq j$ bearbeitet; dabei sei $c_i(S', n, t) > t_0$.

Sei $\bar{t} := \max \{t', t_j\}$.

Nach der Wahl von S' gilt $\bar{t} < t_0$.

Sei S die folgende Strategie:

1. für $[0, \bar{t})$ erzeuge S denselben Ablaufplan wie S' ;

2. für die Zeit nach $\bar{t}$ sind zwei Fälle getrennt zu behandeln:

2.1. Sei $c_j(S', n, t) < c_i(S', n, t)$;

dann lege S fest, daß in allen Zeitintervallen, in denen gemäß S' Auftrag i oder j bearbeitet wird, zunächst j bis zur Fertigstellung, dann i bearbeitet werden soll, während S in allen übrigen Zeitintervallen mit S' übereinstimme.

2.2. Sei $c_i(S', n, t) < c_j(S', n, t)$;
dann lege S fest, daß in allen Zeitintervallen, in denen gemäß S' Auftrag i oder j bearbeitet wird, zunächst i bis zur Fertigstellung, dann j bearbeitet werden soll, während S in allen übrigen Zeitintervallen mit S' übereinstimme.

Nach Konstruktion erlaubt S Ausführungsunterbrechungen höchstens zu den durch T gegebenen Zeitpunkten.

Ebenso folgt $c_i(S, n, t) \leq c_i(S', n, t)$ für alle $i \in \{1, \dots, n\}$.

Nach Def. 2.5. gilt also $\psi(S, n, t) \leq \psi(S', n, t)$. Ist S' eine Strategie, die Ausführungsunterbrechungen in mehreren Zeitpunkten außerhalb T gestattet, so können diese durch wiederholte Anwendung des Konstruktionsverfahrens beseitigt werden. Bei keinem dieser Schritte wächst der Wert bzgl. ψ .

Folgerung: Aus Satz 2.2. folgt, daß es genügt, die Strategien mit Ausführungsunterbrechungen höchstens in den Ankunftszeitpunkten zu untersuchen, wenn eine bzgl. einer regulären Bewertung optimale Strategie gesucht ist. In dem Spezialfall $t_1 = t_2 = \dots = t_n$ genügt es insbesondere Strategien zu untersuchen, die keine Ausführungsunterbrechungen zulassen.

In Satz 2.3. wird angegeben, welche Strategie optimal bzgl. der Bewertung $\bar{v}$ ist, die jeder Strategie die mittlere Verweilzeit eines Auftrages als Wert zuordnet.

Def. 2.8. Seien n und t gegeben mit $t_i \geq 0$ für alle $i \in \{1, \dots, n\}$. Sei

$$e_i(t) := \begin{cases} 0 & \text{für } t \in \mathbb{R}^+ \text{ und } t < t_i; \\ e_i & \text{für } t = t_i; \\ e_i(t') - (t - t') & \text{für } t > t' \geq t_i, \\ & \text{falls Auftrag } i \text{ im Zeitin-} \\ & \text{tervall } [t', t) \text{ bearbeitet} \\ & \text{wird;} \\ e_i(t') & \text{für } t > t' \geq t_i \text{ sonst.} \end{cases}$$

$e_i(t)$ gibt für jedes $i \in \{1, \dots, n\}$ und für jedes $t \geq t_i$ die aktuelle Ausführungsdauer des i -ten Auftrages an.

Satz 2.3. Für beliebige n und t minimiert die Strategie S , die festlegt, daß in jedem Zeitpunkt t der Auftrag $i \in \{1, \dots, n\}$ mit minimaler, positiver aktueller Ausführungsdauer bearbeitet wird, die mittlere Verweilzeit jedes Auftrages.

Bemerkung: Für den Fall, daß zu einem Zeitpunkt mehrere Aufträge die gleiche positive aktuelle Ausführungsdauer haben, kann unter diesen willkürlich aber eindeutig ausgewählt werden, z. B. nach der Auftragsnummer.

Beweis: Seien n und t fest gewählt und S' eine optimale Strategie bzgl. $\bar{v}$. Nach den Sätzen 2.1. und 2.2. kann angenommen werden, daß der von S' erzeugte Ablaufplan keine Leerstelle und Ausführungsunterbrechungen höchstens in den Ankunftszeitpunkten enthält.

Im Gegensatz zur Behauptung des Satzes nehmen wir an, in dem von S' erzeugten Ablaufplan existierten ein Zeitpunkt $t_i^!$ und zwei Aufträge $i, j \in \{1, \dots, n\}$ mit $e_i(t_i^!) > e_j(t_i^!) > 0$; zudem lege S' fest, daß im Zeitintervall $[t_i^!, t_i'']$ der Auftrag i , im Zeitintervall $[t_j^!, t_j'']$ der Auftrag j bearbeitet werde, wobei $t_i^! < t_j^!$ gelte.

Sei S die folgende Strategie:

1. für $[0, t_i^!)$ erzeuge sie denselben Ablaufplan wie S' ;
2. für die späteren Zeitintervalle, in denen gemäß S' die Aufträge i oder j bearbeitet werden, lege sie fest, daß zunächst j bis zur Fertigstellung, dann i bearbeitet werden soll;
3. bzgl. aller übrigen Aufträge erzeuge sie denselben Ablaufplan wie S' .

Sei $c := \max \{c_i(S', n, t), c_j(S', n, t)\}$.

1. Fall: $c = c_i(S', n, t)$.

Dann gelten $c_i(S, n, t) = c_i(S', n, t)$

und $c_j(S, n, t) \leq c_j(S', n, t) - (t_i'' - t_i^!)$.

Sei $c_j(S, n, t) := c_j(S', n, t) - (t_i'' - t_i^!) - \varepsilon$ mit $\varepsilon \geq 0$.

Daraus folgt:

$$\begin{aligned} \bar{v}(S, n, t) &= \frac{1}{n} \sum_{r=1}^n (c_r(S, n, t) - t_r) \\ &= \frac{1}{n} \left\{ \sum_{r \neq i}^n (c_r(S', n, t) - t_r) \right. \\ &\quad \left. + c_j(S', n, t) - (t_i'' - t_i^!) - \varepsilon - t_j \right\} \\ &= \bar{v}(S', n, t) - \frac{1}{n} (\varepsilon + t_i'' - t_i^!) \end{aligned}$$

Wegen $\varepsilon \geq 0$ und $t_i'' > t_i^!$ folgt

$\bar{v}(S, n, t) < \bar{v}(S', n, t)$. Das ist ein Widerspruch zur Annahme S' sei optimal bzgl. $\bar{v}$.

2. Fall: $c = c_j(S', n, t)$.

Dann gelten $c_i(S, n, t) = c_j(S', n, t)$ und

$c_j(S, n, t) \leq c_i(S', n, t) - (e_i(t_i^!) - e_j(t_i^!))$.

Sei $c_j(S, n, t) = c_i(S', n, t) - (e_i(t_i^!) - e_j(t_i^!)) - \varepsilon$ mit $\varepsilon \geq 0$.

Daraus folgt:

$$\begin{aligned}\bar{v}(S, n, t) &= \frac{1}{n} \sum_{r=1}^n (c_r(S, n, t) - t_r) \\ &= \frac{1}{n} \left\{ \sum_{\substack{r=1 \\ r \neq i \\ r \neq j}}^n (c_r(S', n, t) - t_r) \right. \\ &\quad + c_j(S', n, t) - t_i \\ &\quad + c_i(S', n, t) - t_j \\ &\quad \left. - (\varepsilon + e_i(t_i^!) - e_j(t_i^!)) \right\} \\ &= \bar{v}(S', n, t) - \frac{1}{n} (\varepsilon + e_i(t_i^!) - e_j(t_i^!))\end{aligned}$$

Wegen $\varepsilon \geq 0$ und $e_i(t_i^!) > e_j(t_i^!)$ folgt

$$\bar{v}(S, n, t) < \bar{v}(S', n, t).$$

Das ist ein Widerspruch zur Annahme S' sei optimal bzgl. $\bar{v}$.

Der Widerspruchsnachweis wurde unabhängig von speziellen Eigenschaften von n und t geführt, so daß die Aussage für beliebige Aufträge und Ankunftszeitpunkte gilt.

Bemerkung: Es sei darauf hingewiesen, daß die angegebene Strategie auch die mittlere Wartezeit jedes Auftrages minimiert. Nach Def. gilt ja

$$\begin{aligned}\bar{w}(S, n, t) &= \frac{1}{n} \sum_{i=1}^n (c_i(S, n, t) - e_i - t_i) \\ &= \bar{v}(S, n, t) - \frac{1}{n} \sum_{i=1}^n e_i,\end{aligned}$$

so daß sich $\bar{w}$ und $\bar{v}$ bei gegebenen S, n und t um einen konstanten Faktor unterscheiden.

Beispiel:

Wir betrachten insgesamt 4 Aufträge mit

$\pi := (1, 2, 3, 1)$ und $t := (0, 0, 2, 4)$.

Zum Zeitpunkt $t = 0$ sind:

$e_1(0) = 1, e_2(0) = 2, e_3(0) = e_4(0) = 0$.

Im Intervall $[0, 1)$ wird also Auftrag 1 ausgeführt;
also $c_1 = 1$.

Im Zeitintervall $[1, 2)$ wird Auftrag 2 ausgeführt;

zum Zeitpunkt $t = 2$ gelten:

$e_1(2) = 0, e_2(2) = 1, e_3(2) = 3, e_4(2) = 0$.

Folglich wird auch im Zeitintervall $[2, 3)$ der Auftrag 2 ausgeführt und es ergibt sich $c_2 = 3$. Im Zeitintervall $[3, 4)$ wird Auftrag 3 bearbeitet; zum Zeitpunkt $t = 4$ gelten:

$e_1(4) = e_2(4) = 0, e_3(4) = 2, e_4(4) = 1$.

Folglich werden Auftrag 4 in $[4, 5)$ und Auftrag 3 in $[5, 7)$ ausgeführt und es ergeben sich: $c_3 = 7, c_4 = 5$.

Die minimale mittlere Verweilzeit ist

$\bar{v}(\pi, t) = \frac{1}{4}(1+3+5+1) = 2,25$;

die minimale mittlere Wartezeit ist

$\bar{w}(\pi, t) = \frac{1}{4}(0+1+2+0) = 0,75$.

Bild 2 skizziert den Ausführungsablauf.

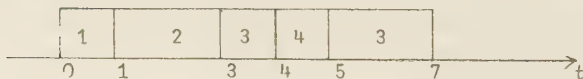


Bild 2: Beispiel für einen Ausführungsablauf

2.1.2. Eine unendliche Folge von Aufträgen

In diesem Abschnitt wird das Problem der Organisation von Programmläufen in dem speziellen Rechenanlagenmodell (universeller Rechnerkern und genügend großer Primärspeicher) als Aufgabe der Warteschlangentheorie formuliert. Neben einigen grundlegenden Definitionen werden Sätze zitiert, die eine gewisse Klasse von Reihenfolgeproblemen lösen.

Es wird angenommen, daß eine unendliche Folge von Aufträgen eingegeben wird. Die Zeit D zwischen der Eingabe aufeinanderfolgender Aufträge sei eine stochastische Variable und genüge einer Verteilungsfunktion $\mathcal{J}(x)$. Analog zu dem Tripel (S, n, t) aus Abschnitt 2.1.1. legt jetzt ein Tripel $(S, \ell(x), \mathcal{J}(x))$ die Ausführung der Programmläufe fest; S ist dabei eine ablaufsteuernde Strategie. Von jeder Strategie nehmen wir an, daß sie die ankommenden Aufträge in Warteschlangen einordnet und nach gewissen Kriterien der Ausführung durch die Bedienungsstation zuführt.

Bei gegebenem S , $\ell(x)$ und $\mathcal{J}(x)$ kann der Ablauf der Auftragsausführungen beschrieben werden durch die Anzahl n der nicht vollständig bearbeiteten Aufträge und die Wahrscheinlichkeit dafür, daß genau $n \in \mathbb{N} \cup \{0\}$ Aufträge vorliegen. Diese Wahrscheinlichkeit ist im allgemeinen eine Funktion der Zeit $t : Q_n(t)$. Statt $Q_n(t)$ interessieren häufig $q_n := \lim_{t \rightarrow \infty} Q_n(t)$, falls diese Grenzwerte existieren. Der Zustand, in dem die Wahrscheinlichkeit für das Anstehen von genau n Aufträgen für alle $n \in \mathbb{N} \cup \{0\}$ zeitunabhängig ist, heißt Gleichgewichtszustand des Systems.

Def. 2.9. Die stochastische Variable D genügt einer Exponentialverteilung mit dem Parameter $\lambda > 0$, wenn für die Verteilungsfunktion $\mathcal{F}(x)$ von D gilt:

$$\mathcal{F}(x) = \begin{cases} 0 & \text{für } x \leq 0 \\ 1 - e^{-\lambda x} & \text{für } x > 0 \end{cases}$$

Genügt D einer Exponentialverteilung mit dem Parameter λ , so ist die Dichte von D $d(x) = \lambda \cdot e^{-\lambda x}$, der Mittelwert $\frac{1}{\lambda}$ und die Varianz $\frac{1}{\lambda^2}$.

Statt der Kennzeichnung des Eingabeprozesses durch D kann die Anzahl $N(t)$ der in einem Zeitintervall der Länge t eingegebenen Aufträge gewählt werden.

Def. 2.10. Die stochastische Variable $N(t)$ genügt einer Poissonverteilung mit dem Parameter $\lambda > 0$, wenn für $n = 0, 1, 2, \dots$ gilt:
Wahrscheinlichkeit $(N(t)=n) = e^{-\lambda t} \cdot \frac{(\lambda t)^n}{n!}$

Genügt $N(t)$ einer Poissonverteilung, so ist der Mittelwert λt und die Varianz λt .

Zwischen der Poissonverteilung und der Exponentialverteilung besteht folgender Zusammenhang:

Satz 2.5. Genügt die Anzahl der Aufträge $N(t)$, die in einem Zeitintervall der Länge t eingegeben werden, einer Poissonverteilung mit dem Parameter $\lambda > 0$ und ist D die Zeit, die zwischen der Eingabe von zwei aufeinanderfolgenden Aufträgen vergeht, so genügt D einer Exponentialverteilung mit dem Parameter λ .

Der Beweis dieses Satzes folgt unmittelbar aus den Definitionen der Poisson- und der Exponentialverteilung.

Die Voraussetzungen dafür, daß ein Eingabeprozess einer Poissonverteilung genügt, sind aus der Wahrscheinlichkeitstheorie wohlbekannt (siehe z. B. Fisz (1965), Kapitel VIII, §3). Es sei bemerkt, daß folgende Aussagen gelten:

Erfolgt eine Eingabe von $n \in \mathbb{N}$ Quellen und von jeder Quelle gemäß einer Poissonverteilung mit dem Parameter λ_i für $i \in \{1, \dots, n\}$, so genügt die Gesamteingabe einer Poissonverteilung mit dem Parameter $\lambda := \sum_{i=1}^n \lambda_i$. Verzweigt sich die Eingabe, die einer Poissonverteilung mit dem Parameter λ genügt, auf $n \in \mathbb{N}$ Wege, und zwar mit der Wahrscheinlichkeit $p_i > 0$ auf den i -ten Weg für $i \in \{1, \dots, n\}$ mit $\sum_{i=1}^n p_i = 1$, so genügt die Eingabe auf jedem dieser Wege einer Poissonverteilung mit dem Parameter $p_i \lambda$. (Diese Aussagen sind z. B. in Conway et al. (1967) S. 144 ff. bewiesen.) Darüber hinaus kann ein Eingabeprozess, der einer Poissonverteilung genügt, auch durch Überlagerung mehrerer Eingabeprozesse, die nicht alle einer Poissonverteilung genügen, gebildet werden.

Für den Fall der Eingabe von Aufträgen gemäß einer Poissonverteilung und unter der Annahme, daß die Ausführungsdauer der Aufträge voneinander unabhängig und gemäß $\ell(x)$ verteilt sind, lassen sich die Probleme der Ablaufsteuerung weitgehend mit den Methoden der Warteschlangentheorie behandeln (siehe z. B. Conway et al. (1967), S. 141 ff.). Wir zitieren hier von zwei Sätzen.

Satz 2.6. Die Eingabe von Aufträgen erfolge gemäß einer Poissonverteilung mit dem Parameter λ und die Dauer eines Programmlaufes sei E mit dem Mittelwert $\bar{E}$ und dem zweiten Moment $\bar{E}^2$. Sei K die Anzahl der anstehenden Aufträge dieses Systems im Gleichgewichtszustand. Dann genügt K für jede Strategie ohne Leerstellen, welche auszuführende Aufträge unabhängig von der Dauer der Programmläufe auswählt, derselben Verteilung mit dem Mittelwert

$$\bar{K} = \lambda \cdot \bar{E} + \frac{\lambda^2 \cdot \bar{E}^2}{2(1 - \lambda \cdot \bar{E})}$$

Der Beweis dieses Satzes ist z. B. in Conway et al. (1967) S. 156 angegeben.

Satz 2.7. Sei V die Verweilzeit eines Auftrages in einem Warteschlangensystem, das sich im Gleichgewichtszustand befindet. Die Eingabe erfolge gemäß einer Poissonverteilung mit dem Parameter λ und $\bar{K}$ sei der Mittelwert der Anzahl der im System anstehenden Aufträge. Dann gilt: $\bar{V} = \frac{1}{\lambda} \cdot \bar{K}$, wobei $\bar{V}$ der Mittelwert von V ist.

Der Beweis dieses Satzes ist z. B. in Little (1961) angegeben.

Die Sätze 2.6. und 2.7. besagen, daß die mittlere Verweilzeit eines Auftrages im System unter den angegebenen Voraussetzungen gleich ist für alle Strategien, die unabhängig von der Dauer der Programmläufe die Reihenfolge festlegen, in der die Aufträge abgearbeitet werden.

Genügt z. B. E einer Exponentialverteilung mit dem Parameter $\mu > 0$, so sind $\bar{E} = \frac{1}{\mu}$ und $\bar{E}^2 = \frac{2}{\mu^2}$, so daß gilt: $\bar{K} = \frac{\lambda}{\mu - \lambda}$ für $\lambda < \mu$; $\lambda < \mu$ ist dabei die Voraussetzung, unter der für das System ein Gleichgewichtszustand existiert. Für die mittlere Verweilzeit ergibt sich: $\bar{V} = \frac{1}{\mu - \lambda}$.

Die Sätze 2.3., 2.6. und 2.7. ergeben insgesamt folgende Aussage für die Steuerung der Programmläufe in einer Rechenanlage, die den Voraussetzungen des Abschnitts 2.1. genügt: Sind von der Ausführungsdauer E der Aufträge ausschließlich der Mittelwert und das zweite Moment bekannt, so daß die Strategie der Ablaufsteuerung sich nicht an der Ausführungsdauer der einzelnen Aufträge orientieren kann, so sind alle Strategien, die keine Leerstellen zulassen bzgl. der mittleren Verweilzeit $\bar{V}$ äquivalent, wenn die Voraussetzungen des Satzes 2.6. erfüllt sind; insbesondere können die Aufträge also in der Reihenfolge ihrer Ankunft ausgeführt werden. Sind hingegen die Ausführungszeiten der Aufträge bekannt, so minimiert die in Satz 2.3. angegebene Strategie die mittlere Verweilzeit jedes Auftrages.

2.2. Zeitmultiplexbetrieb zwischen Rechnerkern und E/A - Steuerwerk

2.2.1. Modellvorstellung

Wir betrachten eine reale Rechenanlage mit einem Rechnerkern und einem E/A - Steuerwerk. Zudem verfüge sie über einen Primärspeicher, dessen Kapazität ausreiche, alle anstehenden Aufträge mit ihren Programmen und Daten aufzunehmen.

Sei $(\Omega, \mathcal{P}(\Omega), P)$ ein Auftrags - Raum. Von den Programmmläufen $\omega \in \Omega$ ist der Speicherraum, der belegt wird, unter den angegebenen Voraussetzungen unwesentlich. Da alle Ein- und Ausgabeoperationen von dem E/A - Steuerwerk ausgeführt werden, sind die in einem Programmmlauf angesprochenen Ein- und Ausgabegeräte verfügbar, wenn das E/A - Steuerwerk verfügbar ist, so daß auch Angaben über diese Geräte in ω vernachlässigt werden können.

Wesentlich an den Programmmläufen ist die Zeit, für die Rechnerkern und E/A - Steuerwerk belegt werden. Jeder Programmmlauf beginne mit einer Aktivität des Rechnerkerns, also mit $\omega(0) = (1, 0, z_3, 0)$ für $z_3 \in \mathbb{Z}_M$. ω belegt den Rechnerkern für eine gewisse Zeit t_1 ; zum Zeitpunkt t_1 kann der Programmmlauf enden, so daß $\omega(t_1) = (0, 0, 0, 0)$ und $\omega(t) \neq (0, 0, 0, 0)$ für $0 \leq t < t_1$ gelten; andererseits kann der Programmmlauf vom Zeitpunkt t_1 an im E/A - Steuerwerk weitergeführt werden und es gilt für $0 \leq t < t_1$ $\omega(t) = (1, 0, z_3, 0)$, während $\omega(t_1) = (0, 1, z_3, z_4)$. In diesem Fall belegt ω das E/A - Steuerwerk für eine gewisse Zeit t_2 ; zum Zeitpunkt $t_1 + t_2$ kann der Programmmlauf enden, so daß $\omega(t_1 + t_2) = (0, 0, 0, 0)$ und $\omega(t) \neq (0, 0, 0, 0)$ für $0 \leq t < t_1 + t_2$ gelten. Statt dessen kann der Programmmlauf vom Zeitpunkt $t_1 + t_2$ wieder vom Rechnerkern gesteuert werden und es gelten $\omega(t) = (0, 1, z_3, z_4)$ für $t_1 \leq t < t_1 + t_2$, und $\omega(t_1 + t_2) = (1, 0, z_3, 0)$. In dieser Art fortgeführt ergibt sich für einen speziellen Programmmlauf ω eine Folge von $n \in \mathbb{N}$ Zeitintervallen $[0, t_1), [t_1, t_1 + t_2), \dots, [\sum_{i=1}^{n-1} t_i, \sum_{i=1}^n t_i)$ derart, daß für ω gelten:

1. Sei $T := \sum_{i=1}^n t_i$; dann gilt $\omega(T) = (0, 0, 0, 0)$ und $\omega(t) \neq (0, 0, 0, 0)$ für alle t mit $0 \leq t < T$.

2. Für alle Zeitpunkte t aus dem 1., 3.,... Zeitintervall gelten $\omega(t) = (1, 0, z_3, 0)$.
3. Für alle Zeitpunkte t aus dem 2., 4.,... Zeitintervall gelten $\omega(t) = (0, 1, z_3, z_4)$.

Die Gesamtheit der Programmläufe Ω ergibt eine analoge Beschreibung unter Berücksichtigung der Wahrscheinlichkeiten, mit denen spezielle Folgen von Zeitintervallen auftreten. Da z. Z. Meßwerte fehlen, die für einen bestimmten Auftrags - Raum entsprechende Aussagen ermöglichen, sei ein Auftrags - Raum angenommen, der sich wie folgt für die speziellen Bedürfnisse dieses Abschnitts beschreiben läßt:

1. Jeder Programmlauf beginnt im Rechnerkern.
2. Endet eine Programmlauf - Phase im Rechnerkern, so endet mit dieser Phase der Programmlauf mit der Wahrscheinlichkeit p_1 , während sich mit der Wahrscheinlichkeit $1 - p_1$ eine Programmlauf - Phase im E/A - Steuerwerk anschließt.
3. Endet eine Programmlauf - Phase im E/A - Steuerwerk, so endet mit dieser Phase der Programmlauf mit der Wahrscheinlichkeit p_2 , während sich mit der Wahrscheinlichkeit $1 - p_2$ eine Programmlauf - Phase im Rechnerkern anschließt.
4. Sei E_1 die Dauer einer Programmlauf - Phase im Rechnerkern; E_1 ist stochastische Variable und genüge einer Verteilungsfunktion $f_1(x)$.
5. Sei E_2 die Dauer einer Programmlauf - Phase im E/A - Steuerwerk; E_2 ist stochastische Variable und genüge einer Verteilungsfunktion $f_2(x)$. E_1 und E_2 seien unabhängig voneinander.

Bild 3 skizziert einen Programmlauf unter diesen Annahmen.

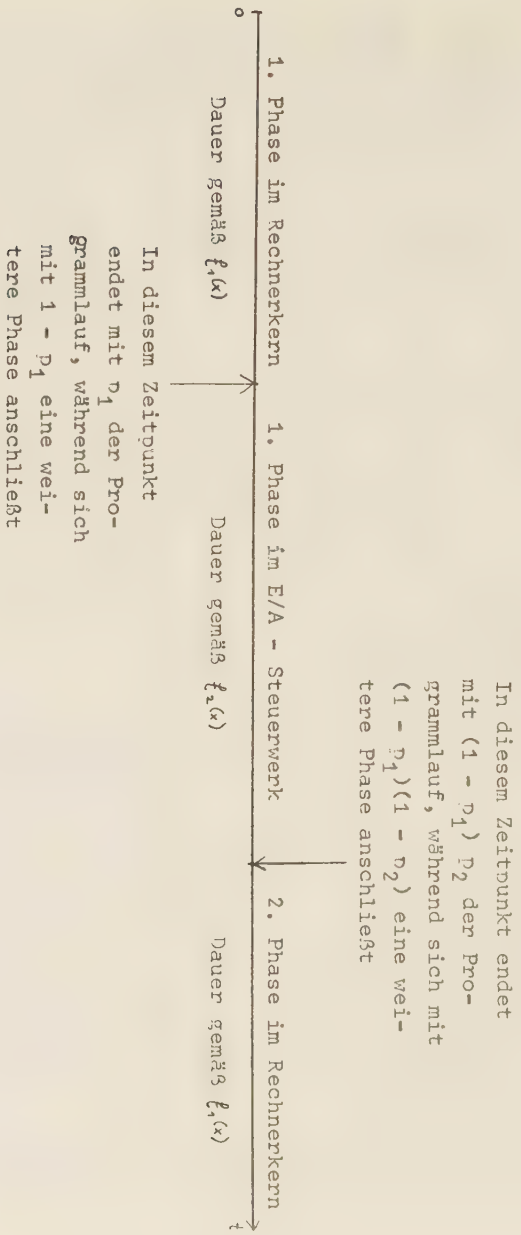


Bild 3: Programmläufe in Rechnerkern und E/A - Steuerwerk

Für spezielle $\xi_1(x)$ und $\xi_2(x)$ wird in Abschnitt 2.2.2. ein Warteschlangensystem untersucht. In Abschnitt 2.2.3. wird dieses System, das Zeitmultiplexbetrieb gestattet, verglichen mit zwei weiteren Systemen.

2.2.2. Ein Warteschlangensystem für Zeitmultiplexbetrieb

Zur Analyse der Programmläufe in einer Rechenanlage mit der angegebenen Struktur betrachten wir ein Warteschlangensystem mit folgenden Eigenschaften:

1. Das System verfügt über zwei Bedienstationen, von denen die eine den Rechnerkern, die andere das E/A - Steuerwerk repräsentieren soll.
2. Die Eingabe der Aufträge erfolge gemäß einer Poissonverteilung mit dem Parameter $\lambda > 0$.
3. Jeder Auftrag wird bei seiner Eingabe in die Warteschlange des Rechnerkerns eingeordnet.
4. Die Ausführungsdauer im Rechnerkern genüge einer Exponentialverteilung mit dem Parameter μ_1 .
5. Nach Abschluß der Ausführung im Rechnerkern verlasse der Auftrag mit der Wahrscheinlichkeit p_1 das System, während er mit $1 - p_1$ in die Warteschlange des E/A - Steuerwerkes eingeordnet werde.
6. Die Ausführungsdauer im E/A - Steuerwerk genüge einer Exponentialverteilung mit dem Parameter μ_2 .
7. Nach Abschluß der Ausführung im E/A - Steuerwerk verlasse der Auftrag mit der Wahrscheinlichkeit p_2 das System, während er mit $1 - p_2$ in die Warteschlange des Rechnerkerns eingeordnet werde.
8. Beide Warteschlangen seien in ihrer Länge unbeschränkt; jede Warteschlange werde in der Reihenfolge der Ankünfte abgearbeitet (Strategie FIFO).

Bild 4 skizziert das Warteschlangensystem.

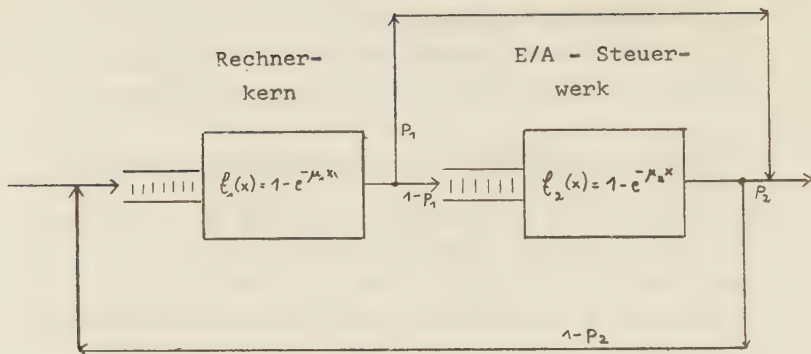


Bild 4: Ein Warteschlangensystem zur Analyse des Zeitmultiplexbetriebes

Bevor die Anzahl der Aufträge in diesem System berechnet wird, werden zwei Eigenschaften der Exponentialverteilung angegeben.

Satz 2.8. Sei D die Zeit, die zwischen der Eingabe aufeinanderfolgender Aufträge vergeht und D genüge einer Exponentialverteilung mit dem Parameter λ . Seien t' und t'' Zeitpunkte der Eingabe aufeinanderfolgender Aufträge. Dann gilt für $t' < t < t''$ und $x > 0$ unabhängig von t : Wahrscheinlichkeit $(D < t+x | D \geq t) = 1 - e^{-\lambda x}$.

Beweis: Es gilt Wahrscheinlichkeit $(D < t+x | D \geq t) =$

$$= \frac{\text{Wahrscheinlichkeit } (t \leq D < t+x)}{\text{Wahrscheinlichkeit } (D \geq t)}$$

Aus der Voraussetzung über D folgt:

$$= \frac{\int_0^{t+x} \lambda \cdot e^{-\lambda \tau} d\tau}{\int_0^{\infty} \lambda \cdot e^{-\lambda \tau} d\tau} = 1 - e^{-\lambda x}.$$

Dieser Satz bringt die "Gedächtnislosigkeit" der Exponentialverteilung zum Ausdruck; er besagt, daß die Wahrscheinlichkeit für das Eintreffen eines Auftrages in einem Zeitintervall $[t, t+x)$ ausschließlich von der Länge des Zeitintervalles abhängig ist.

Satz 2.9. Sei $N(t)$ die Anzahl der Aufträge, die in einem Zeitintervall der Länge t eingegeben werden und $N(t)$ genüge einer Poissonverteilung mit dem Parameter λ . Dann gelten für $\Delta t > 0$ und $\Delta t \rightarrow 0$:

1. Wahrscheinlichkeit

$$(N(t)=0) = e^{-\lambda \cdot \Delta t} \sim 1 - \lambda \cdot \Delta t + o(\Delta t^2);$$

d. h. die Wahrscheinlichkeit, daß in einem Zeitintervall $[t, t+\Delta t)$ kein Auftrag eingegeben wird, verhält sich für $\Delta t \rightarrow 0$ wie $1 - \lambda \cdot \Delta t + o(\Delta t^2)$.

2. Wahrscheinlichkeit

$$(N(t)=1) = \lambda \cdot e^{-\lambda \cdot \Delta t} \cdot \Delta t \sim \lambda \cdot \Delta t + o(\Delta t^2);$$

d. h. die Wahrscheinlichkeit, daß in einem Zeitintervall $[t, t+\Delta t)$ genau ein Auftrag eingegeben wird, verhält sich für $\Delta t \rightarrow 0$ wie $\lambda \cdot \Delta t + o(\Delta t^2)$.

3. Wahrscheinlichkeit

$$(N(t) \geq 2) = 1 - e^{-\lambda \cdot \Delta t} - \lambda \cdot e^{-\lambda \cdot \Delta t} \cdot \Delta t \sim o(\Delta t^2);$$

d. h. die Wahrscheinlichkeit, daß in einem Zeitintervall $[t, t+\Delta t)$ wenigstens zwei Aufträge eingegeben werden, verhält sich für $\Delta t \rightarrow 0$ wie $o(\Delta t^2)$.

Der Beweis dieser Aussagen ergibt sich unmittelbar aus der Definition der Poissonverteilung und der Taylor - Reihe der Exponentialfunktion.

Nach diesen Vorbemerkungen kann das beschriebene Warteschlangen - System analysiert werden.

Satz 2.10. Gegeben sei ein Warteschlangensystem mit den Eigenschaften 1. bis 8. dieses Abschnittes. Sind $K_1(t)$ bzw. $K_2(t)$ die Anzahl der Aufträge, die zum Zeitpunkt t in der Warteschlange des Rechnerkerns bzw. des E/A - Steuerwerkes anstehen, $p(n,m,t) :=$ Wahrscheinlichkeit ($K_1(t)=n$ und $K_2(t)=m$) und $p(n,m) = \lim_{t \rightarrow \infty} p(n,m,t)$ für $n,m \in \{0,1,2,\dots\}$, so gelten:

1. Unter den Voraussetzungen

$$1.1. \quad p_1 + p_2 - p_1 \cdot p_2 \neq 0$$

$$1.2. \quad \frac{\lambda}{\mu_1} < p_1 + p_2 - p_1 \cdot p_2$$

$$1.3. \quad \frac{\lambda}{\mu_2} \cdot (1 - p_1) < p_1 + p_2 - p_1 \cdot p_2$$

existiert ein Gleichgewichtszustand des Systems.

2. Im Gleichgewichtszustand sind

$$2.1. \quad p(n,m) = a^n \cdot b^m \cdot (1-a) \cdot (1-b)$$

$$\text{mit } a = \frac{\lambda}{\mu_1 \cdot (p_1 + p_2 - p_1 \cdot p_2)}$$

$$\text{und } b = \frac{\lambda \cdot (1 - p_1)}{\mu_2 \cdot (p_1 + p_2 - p_1 \cdot p_2)}$$

$$2.2. \quad \text{Der Mittelwert von } K_1 \text{ ist } \bar{K}_1 = \frac{a}{1-a}$$

$$2.3. \quad \text{Der Mittelwert von } K_2 \text{ ist } \bar{K}_2 = \frac{b}{1-b}$$

Beweis: Der Beweis wird in folgenden Schritten geführt:

1. Für die $p(n,m,t)$ wird ein Differentialgleichungssystem angegeben.

2. Für zeitunabhängige Gleichgewichtslösungen gelten $\frac{\partial p(n,m,t)}{\partial t} = 0$. Somit ergibt sich aus dem Differentialgleichungssystem ein Gleichungssystem für die $p(n,m)$. Dieses wird gelöst und liefert die angegebenen Aussagen.

1. Schritt:

Seien $t \geq 0$ und $\Delta t > 0$. Alle nachfolgenden Differenzengleichungen ergeben sich durch Anwendung der Sätze 2.8. und 2.9. auf die Eingabe und die Ausführung der Aufträge. Die Wahrscheinlichkeiten $p(n,m,t+\Delta t)$ werden berechnet, indem alle Fälle berücksichtigt werden, die in einem Zeitintervall der Länge Δt mit positiver Wahrscheinlichkeit einen Übergang in den Zustand $K_1=n$ und $K_2=m$ erlauben.

1.1. $K_1 = 0$ und $K_2 = 0$:

$$\begin{aligned} p(0,0,t+\Delta t) = & p(0,0,t)(1-\lambda \cdot \Delta t) + \\ & + p(1,0,t)(1-\lambda \cdot \Delta t) \mu_1 \cdot \Delta t \cdot p_1 + \\ & + p(0,1,t)(1-\lambda \cdot \Delta t) \mu_2 \cdot \Delta t \cdot p_2 + o(\Delta t^2) \end{aligned}$$

1.2. $K_1 = 1$ und $K_2 = 0$:

$$\begin{aligned} p(1,0,t+\Delta t) = & p(1,0,t)(1-\lambda \cdot \Delta t)(1-\mu_1 \cdot \Delta t) \\ & + p(0,0,t) \lambda \cdot \Delta t + \\ & + p(0,1,t)(1-\lambda \Delta t) \mu_2 \cdot \Delta t (1-p_2) \\ & + p(2,0,t)(1-\lambda \Delta t) \mu_1 \cdot \Delta t \cdot p_1 \\ & + p(1,1,t)(1-\lambda \Delta t)(1-\mu_1 \cdot \Delta t) \mu_1 \cdot \Delta t \cdot p_2 + o(\Delta t^2) \end{aligned}$$

1.3. $n \geq 2$ und $K_2 = 0$:

$$\begin{aligned} p(n,0,t+\Delta t) = & p(n,0,t)(1-\lambda \cdot \Delta t)(1-\mu_1 \cdot \Delta t) + \\ & + p(n-1,0,t) \lambda \Delta t (1-\mu_1 \cdot \Delta t) + \\ & + p(n-1,1,t)(1-\lambda \Delta t)(1-\mu_1 \cdot \Delta t) \mu_1 \Delta t (1-p_2) + \\ & + p(n+1,0,t)(1-\lambda \Delta t) \mu_1 \cdot \Delta t \cdot p_1 + \\ & + p(n,1,t)(1-\lambda \Delta t)(1-\mu_1 \cdot \Delta t) \mu_1 \cdot \Delta t p_2 + o(\Delta t^2) \end{aligned}$$

1.4. $K_1 = 0$ und $K_2 = 1$:

$$\begin{aligned} p(0,1,t+\Delta t) &= p(0,1,t)(1-\lambda\Delta t)(1-\mu_1\Delta t) + \\ &+ p(1,1,t)(1-\lambda\Delta t)\mu_1\Delta t p_1(1-\mu_1\Delta t) + \\ &+ p(1,0,t)(1-\lambda\Delta t)\mu_1\Delta t(1-p_1) + \\ &+ p(0,2,t)(1-\lambda\Delta t)\mu_1\Delta t p_2 + o(\Delta t^2) \end{aligned}$$

1.5. $K_1 = 0$ und $m \geq 2$:

$$\begin{aligned} p(0,m,t+\Delta t) &= p(0,m,t)(1-\lambda\Delta t)(1-\mu_1\Delta t) + \\ &+ p(1,m,t)(1-\lambda\Delta t)\mu_1\Delta t p_1(1-\mu_1\Delta t) + \\ &+ p(1,m-1,t)(1-\lambda\Delta t)\mu_1\Delta t(1-p_1)(1-\mu_1\Delta t) + \\ &+ p(0,m+1,t)(1-\lambda\Delta t)\mu_1\Delta t p_2 + o(\Delta t^2) \end{aligned}$$

1.6. $K_1 = 1$ und $K_2 = 1$:

$$\begin{aligned} p(1,1,t+\Delta t) &= p(1,1,t)(1-\lambda\Delta t)(1-\mu_1\Delta t)(1-\mu_2\Delta t) + \\ &+ p(0,1,t)\lambda\Delta t(1-\mu_1\Delta t) + \\ &+ p(0,2,t)(1-\lambda\Delta t)\mu_2\Delta t(1-p_2) + \\ &+ p(2,1,t)(1-\lambda\Delta t)\mu_1\Delta t p_1(1-\mu_1\Delta t) + \\ &+ p(2,0,t)(1-\lambda\Delta t)\mu_1\Delta t(1-p_1) + \\ &+ p(1,2,t)(1-\lambda\Delta t)(1-\mu_1\Delta t)\mu_2\Delta t p_2 + o(\Delta t^2) \end{aligned}$$

1.7. $n \geq 2$ und $m \geq 2$:

$$\begin{aligned} p(n,m,t+\Delta t) &= p(n,m,t)(1-\lambda\Delta t)(1-\mu_1\Delta t)(1-\mu_2\Delta t) + \\ &+ p(n-1,m,t)\lambda\Delta t(1-\mu_1\Delta t)(1-\mu_2\Delta t) + \\ &+ p(n-1,m+1,t)(1-\lambda\Delta t)(1-\mu_1\Delta t)\mu_2\Delta t(1-p_2) + \\ &+ p(n+1,m,t)(1-\lambda\Delta t)\mu_1\Delta t p_1(1-\mu_1\Delta t) + \\ &+ p(n+1,m-1,t)(1-\lambda\Delta t)\mu_1\Delta t(1-p_1)(1-\mu_1\Delta t) + \\ &+ p(n,m+1,t)(1-\lambda\Delta t)(1-\mu_1\Delta t)\mu_2\Delta t p_2 + o(\Delta t^2) \end{aligned}$$

Daraus ergibt sich nach Umordnung und $\Delta t \rightarrow 0$:

$$\begin{aligned} p'(0,0,t) &= -\lambda \cdot p(0,0,t) + p_1 \mu_1 \cdot p(1,0,t) \\ &+ p_2 \mu_1 \cdot p(0,1,t) \end{aligned}$$

$$\begin{aligned} p'(n,0,t) &= -(\lambda + \mu_1) p(n,0,t) + \lambda \cdot p(n-1,0,t) \\ &+ (1-p_2) \mu_2 p(n-1,1,t) \\ &+ p_1 \mu_1 \cdot p(n+1,0,t) + p_2 \mu_2 p(n,1,t) \end{aligned}$$

für $n \geq 1$

$$\begin{aligned} p'(0,m,t) &= -(\lambda + \mu_2) p(0,m,t) + p_1 \mu_2 \cdot p(1,m,t) \\ &+ (1-p_1) \mu_1 \cdot p(1,m-1,t) \\ &+ p_2 \mu_2 \cdot p(0,m+1,t) \end{aligned}$$

für $m \geq 1$

$$\begin{aligned}
 p'(n,m,t) = & -(\lambda + \mu_1 + \mu_2) p(n,m,t) \\
 & + \lambda \cdot p(n-1,m,t) \\
 & + (1-p_2) \cdot \mu_1 \cdot p(n-1,m+1,t) \\
 & + p_1 \cdot \mu_1 \cdot p(n+1,m,t) \\
 & + (1-p_1) \cdot \mu_1 \cdot p(n+1,m-1,t) \\
 & + p_2 \cdot \mu_2 \cdot p(n,m+1,t) \\
 & \text{für } n \geq 1 \text{ und } m \geq 1.
 \end{aligned}$$

Dabei ist $p'(n,m,t) := \frac{\partial p(n,m,t)}{\partial t}$

2. Schritt:

Im Gleichgewichtszustand gilt $p'(n,m,t)=0$ für $n,m \in \{0,1,\dots\}$. Durch Einsetzen ergibt sich ein Gleichungssystem für $p(n,m)$.

$$\begin{aligned}
 \lambda \cdot p(0,0) &= p_1 \cdot \mu_1 \cdot p(1,0) + p_2 \cdot \mu_2 \cdot p(0,1) \\
 (\lambda + \mu_1) p(n,0) &= \lambda \cdot p(n-1,0) + (1-p_2) \cdot \mu_1 \cdot p(n-1,1) \\
 &\quad + p_1 \cdot \mu_1 \cdot p(n+1,0) + p_2 \cdot \mu_2 \cdot p(n,1) \text{ für } n \geq 1 \\
 (\lambda + \mu_2) p(0,m) &= p_1 \cdot \mu_1 \cdot p(1,m) + (1-p_1) \cdot \mu_1 \cdot p(1,m-1) \\
 &\quad + p_2 \cdot \mu_2 \cdot p(0,m+1) \text{ für } m \geq 1 \\
 (\lambda + \mu_1 + \mu_2) p(n,m) &= \lambda \cdot p(n-1,m) + (1-p_2) \cdot \mu_1 \cdot p(n-1,m+1) \\
 &\quad + p_1 \cdot \mu_1 \cdot p(n+1,m) \\
 &\quad + (1-p_1) \cdot \mu_1 \cdot p(n+1,m-1) \\
 &\quad + p_2 \cdot \mu_2 \cdot p(n,m+1) \text{ für } n \geq 1 \text{ und } m \geq 1.
 \end{aligned}$$

Da $p(n,m)$ für $n,m \in \{0,1,\dots\}$ Wahrscheinlichkeiten eines vollständigen Ereignissystems sind, gilt zusätzlich $\sum_{n=0}^{\infty} \sum_{m=0}^{\infty} p(n,m) = 1$

Seien $p_1 + p_2 - p_1 p_2 \neq 0$

$$\text{und } a := \frac{\lambda}{\mu_1 \cdot (p_1 + p_2 - p_1 p_2)}$$

$$b := \frac{\lambda \cdot (1-p_1)}{\mu_2 \cdot (p_1 + p_2 - p_1 p_2)}$$

Für $a < 1$ und $b < 1$ hat das angegebene Gleichungssystem die Lösungen $p(n,m) = a^n \cdot b^m \cdot (1-a)(1-b)$; dies ergibt sich durch Einsetzen in die Gleichungen.

Für die Randverteilungen von K_1 und K_2 gelten:

$$p(n, \cdot) = \sum_{m=0}^{\infty} p(n, m) = a^n (1-a) \text{ und}$$

$$p(\cdot, m) = \sum_{n=0}^{\infty} p(n, m) = b^m (1-b)$$

Daraus ergeben sich für die Mittelwerte:

$$\bar{K}_1 = \sum_{n=0}^{\infty} n \cdot p(n, \cdot) = \frac{a}{1-a} \text{ und}$$

$$\bar{K}_2 = \sum_{m=0}^{\infty} m \cdot p(\cdot, m) = \frac{b}{1-b}.$$

Bemerkungen:

1. Die Voraussetzungen für die Existenz eines Gleichgewichtszustandes lassen sich wie folgt deuten:

Zu Voraussetzung 1.1.

Für $0 \leq p_1, p_2 \leq 1$ gilt $0 \leq p_1 + p_2 - p_1 p_2 \leq 1$ und

$p_1 + p_2 - p_1 p_2 = 0$ für $p_1 = p_2 = 0$. Das bedeutet für die Programmläufe, daß sie mit der Wahrscheinlichkeit 1 niemals das System verlassen.

Zu den Voraussetzungen 1.2. und 1.3.

In diesen Voraussetzungen wird der Grad der Verkoppelung von Operationen des Rechnerkerns und E/A - Vorgängen berücksichtigt. In dem Sonderfall $p_1 = 1$ ergibt sich aus 1.2. die Forderung $\frac{\lambda}{\mu_1} < 1$; also bestimmt der Rechnerkern das Verhalten des Systems. Für $p_1 = 0$ und $p_2 = 1$ ergeben sich 1.2. zu $\frac{\lambda}{\mu_1} < 1$ und 1.3. zu $\frac{\lambda}{\mu_1} < 1$. In diesem Fall belegt jeder Programmlauf Rechnerkern und E/A - Kanal genau einmal für die Dauer einer Ausführungs - Phase; der Gleichgewichtszustand existiert, wenn für Rechnerkern und E/A - Steuerwerk die mittlere Zeitdauer zwischen zwei Eingaben kleiner ist, als die mittlere Ausführungsdauer eines Auftrages.

2. Sei $K := K_1 + K_2$; dann ergibt sich für den Mittelwert: $\bar{K} = \bar{K}_1 + \bar{K}_2$, also $\bar{K} = \frac{a}{1-a} + \frac{b}{1-b}$. Damit ist für den Gleichgewichtszustand der Mittelwert der Anzahl von Aufträgen, die unausgeführt sind, bekannt.

Für das Folgende benötigen wir die Laplace - Transformierte von Verteilungsfunktionen und einige ihrer Eigenschaften.

Def. 2.9. Ist $\xi(x)$ Verteilungsfunktion einer stochastischen Veränderlichen E mit $\xi(x)=0$ für $x < 0$ und der Dichte $f(x)$, so ist

$$\varphi(z) := \int_0^{\infty} e^{-zx} \cdot f(x) dx$$

die Laplace - Transformierte von $f(x)$.
 $\varphi(z)$ ist Funktion der komplexwertigen Variablen z .

Bemerkungen:

1. Ist $\xi(x)=1-e^{-\lambda x}$ für $x \geq 0$, so gilt
- $$\varphi(z) = \int_0^{\infty} e^{-zx} \cdot \lambda \cdot e^{-\lambda x} dx = \frac{\lambda}{z+\lambda}$$
- und $\varphi(z)$ existiert in der Halbebene der z - Werte mit $\operatorname{Re}(z) > -\lambda$.
2. Sind E_1 und E_2 unabhängige stochastische Variable mit den Laplace - Transformaten $\varphi_1(z)$ und $\varphi_2(z)$ ihrer Dichten. Ist $E := E_1 + E_2$ und $\varphi(z)$ Laplace - Transformierte der Dichte von E, so gilt:
- $$\varphi(z) = \varphi_1(z) \cdot \varphi_2(z).$$
- Für die Existenz von $\varphi(z)$ gilt: Konvergieren die Laplace - Integrale bzgl. $\varphi_1(z)$ und $\varphi_2(z)$ für z mit $\operatorname{Re}(z) \geq a$ absolut, so konvergiert in dieser Halbebene auch $\varphi(z)$ absolut.

Nach diesen Vorbemerkungen kann eine Aussage über die Dauer der Programmläufe angegeben werden, die dem in Abschnitt 2.2.1. beschriebenen Auftrags - Raum angehören.

Satz 2.11. Ein Programm - Raum habe die Eigenschaften

1. bis 5. von Abschnitt 2.2.1. $\varphi_1(z)$ sei die L-Transformierte der Dichte von E_1 ,

$q_1(z)$ die L-Transformierte der Dichte von E_2 . Ist $f(x)$ die Verteilungsfunktion der Dauer E der Programmläufe und $q(z)$ die L-Transformierte ihrer Dichte, so gelten:

1. Falls $p_1 = 1$: $q(z) = q_1(z)$.
2. Falls $0 \leq p_1 < 1$ und $p_2 = 1$:

$$q(z) = p_1 \cdot q_1(z) + (1-p_1) p_2 \cdot q_1(z) \cdot q_1(z)$$
3. Falls $0 < p_1, p_2 < 1$ und

$$|(1-p_1)(1-p_2) \cdot q_1(z) \cdot q_1(z)| < 1$$

$$q(z) = \frac{(p_1 + (1-p_1)p_2 \cdot q_1(z)) \cdot q_1(z)}{1 - (1-p_1-p_2+p_1p_2) \cdot q_1(z) \cdot q_1(z)}$$
4. Genügen speziell E_1 und E_2 Exponentialverteilungen mit den Parametern μ_1 und μ_2 , so genügt E für $0 < p_1, p_2 < 1$ einer Exponentialverteilung mit dem Parameter μ , falls $p_1 \cdot \mu_1 = p_2 \cdot \mu_2$ gilt; dabei ist $\mu = p_1 \cdot \mu_1$.

Beweis: 1. Falls $p_1 = 1$ gilt, folgt $E = E_1$, also:

$$q(z) = q_1(z).$$

2. Falls $0 \leq p_1 < 1$ und $p_2 = 1$ gilt, endet der Programmlauf mit p_1 nach der 1. Rechnerkern-Phase und mit $(1-p_1)$ nach der 1. E/A - Steuerwerk - Phase. Daraus folgt:

$$q(z) = p_1 \cdot q_1(z) + (1-p_1) \cdot q_1(z) \cdot q_1(z)$$

3. Falls $0 < p_1, p_2 < 1$ gelten, setzt sich E aus $n \in \mathbb{N}$ Phasen zusammen, deren Längen unabhängig voneinander und gemäß $f_1(x)$ bzw. $f_2(x)$ verteilt sind. Also gilt:

$$\begin{aligned} q(z) &= p_1 \cdot q_1(z) + (1-p_1) \cdot p_2 \cdot q_1(z) \cdot q_1(z) + \\ &\quad (1-p_1)(1-p_2)p_1(q_1(z))^2 \cdot q_1(z) + \\ &\quad (1-p_1)^2(1-p_2)p_2 \cdot q_1^2(z) \cdot q_1^2(z) + \dots \\ &= \sum_{n=0}^{\infty} (1-p_1)^n (1-p_2)^n p_1 \cdot q_1^{n+1}(z) \cdot q_1^n(z) \\ &\quad + \sum_{n=0}^{\infty} (1-p_1)^{n+1} (1-p_2)^n \cdot p_2 \cdot q_1^{n+1}(z) \cdot q_1^{n+1}(z) \end{aligned}$$

Falls $|(1-p_1)(1-p_2)\varrho_1(z)\varrho_2(z)| < 1$ gilt, folgt

$$\varrho(z) = \frac{(p_1 + (1-p_1)p_2\varrho_1(z))\varrho_1(z)}{1 - (1-p_1-p_2+p_1p_2)\varrho_1(z)\varrho_2(z)}$$

4. Genügen E_1 und E_2 Exponentialverteilungen mit den Parametern μ_1 und μ_2 , so sind

$$\varrho_1(z) = \frac{\mu_1}{z + \mu_1} \text{ und } \varrho_2(z) = \frac{\mu_2}{z + \mu_2} \text{ und beide Laplace -}$$

Integrale konvergieren absolut für

$$\operatorname{Re}(z) > -\min\{\mu_1, \mu_2\}.$$

$$\text{Sei } \mu := p_1\mu_1 = p_2\mu_2$$

Beh: Für $\operatorname{Re}(z) > -\mu$ gilt

$$|(1-p_1)(1-p_2)\varrho_1(z)\varrho_2(z)| < 1$$

$$\text{Bew: } |(1-p_1)(1-p_2)\varrho_1(z)\varrho_2(z)| < 1$$

$$(1-p_1)(1-p_2)\mu_1\mu_2 < |(z+\mu_1)(z+\mu_2)|$$

Sei $z' := -\mu + \varepsilon$ mit $\varepsilon > 0$. Für alle z mit

$$\operatorname{Re}(z) > -\mu \text{ gelten } |z+\mu_1| \geq -\mu + \varepsilon + \mu_1 \text{ und}$$

$$|z+\mu_2| \geq -\mu + \varepsilon + \mu_2. \text{ Daraus folgt:}$$

$$|(z+\mu_1)(z+\mu_2)| \geq (\mu_1 + \varepsilon - \mu)(\mu_2 + \varepsilon - \mu)$$

$$= (1-p_1)(1-p_2)\mu_1\mu_2$$

$$+ \varepsilon \underbrace{(\mu_1(1-p_1) + \mu_2(1-p_2) + \varepsilon)}_{> 0}$$

Für $\operatorname{Re}(z) > -\mu$ gilt also nach 3.

$$\varrho(z) = \frac{(p_1 + (1-p_1)p_2\varrho_1(z))\varrho_1(z)}{1 - (1-p_1-p_2+p_1p_2)\varrho_1(z)\varrho_2(z)} = \frac{\mu}{z + \mu}$$

2.2.3. Strukturvergleiche

In diesem Abschnitt werden drei in ihrer Struktur verschiedene Rechenanlagenmodelle miteinander verglichen. In jede Anlage werden Aufträge gemäß einer Poissonverteilung mit dem Parameter λ eingegeben und die Dauer der Programmläufe genüge jeweils einer Exponentialverteilung mit dem Parameter μ . Dem Vergleich der Anlagen wird die mittlere Anzahl der unausgeführten

Aufträge im Gleichgewichtszustand des Systems zugrunde gelegt.

Modell 1 enthält einen universellen Rechnerkern (wie in 2.1. beschrieben), der alle Programmläufe steuert. Die Aufträge werden in der Reihenfolge ihrer Eingabe abgearbeitet.

Modell 2 sei das in Abschnitt 2.2.2. behandelte System.

Modell 3 enthalte zwei funktionsäquivalente, universelle Rechnerkerne. Die Aufträge werden in der Reihenfolge ihrer Eingabe ausgeführt, und zwar von einem beliebigen der beiden Rechnerkerne.

Bild 5 skizziert die drei Modelle.

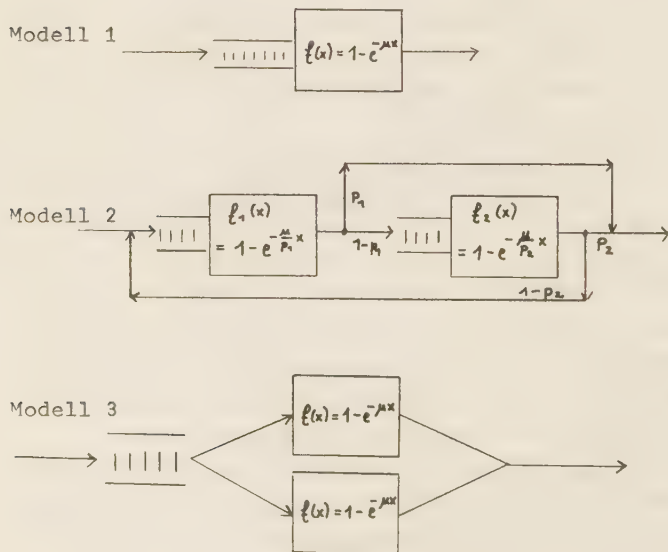


Bild 5: Die Struktur der miteinander verglichenen Systeme

Sei K_1 die Anzahl der unausgeführten Aufträge im Gleichgewichtszustand des Modelles 1. Der Gleichgewichtszustand existiert für $\lambda < \mu$ und nach einer Anmerkung in Abschnitt 2.1.2. gilt für den Mittelwert $\bar{K}_1$ von K_1 : $\bar{K}_1 = \frac{\lambda}{\mu - \lambda}$.

Sei K_2 die Anzahl der unausgeführten Aufträge im Gleichgewichtszustand des Modelles 2. Nach den Sätzen 2.10. und 2.11. existiert für $0 < p_1, p_2 < 1$ ein Gleichgewichtszustand, falls $\frac{\lambda}{\mu} p_1 < p_1 + p_2 - p_1 p_2$ und $\frac{\lambda}{\mu} p_2 < p_1 + p_2 - p_2 p_1$ gelten. Dann gilt für den Mittelwert $\bar{K}_2$ von K_2 :

$$\bar{K}_2 = \frac{\bar{a}}{1-a} + \frac{\bar{b}}{1-b} \quad \text{mit } \bar{a} := \frac{\lambda \cdot p_1}{\mu(p_1 + p_2 - p_1 p_2)} \quad \text{und} \\ \bar{b} := \frac{\lambda(1-p_1)p_2}{\mu(p_1 + p_2 - p_1 p_2)}.$$

Sei K_3 die Anzahl der unausgeführten Aufträge im Gleichgewichtszustand des Modelles 3. Nach Saaty (1961) gilt für dieses: Ein Gleichgewichtszustand des Systems existiert für $\lambda < 2\mu$ und für den Mittelwert $\bar{K}_3$ von K_3 gilt:

$$\bar{K}_3 = \frac{4\lambda\mu}{4\mu^2 - \lambda^2}$$

Für das Modell 2 ist $\bar{K}_2$ eine Funktion von p_1 und p_2 . Der folgende Satz faßt den Einfluß von p_1 und p_2 auf $\bar{K}_2$ in einem Vergleich der drei Modelle bei festen Werten von λ und μ zusammen.

Satz 2.12. Sind K_1, K_2 bzw. K_3 die Anzahl der unausgeführten Aufträge der Modelle 1, 2 bzw. 3 im Gleichgewichtszustand mit $\lambda < \mu$ und $0 < p_1, p_2 < 1$, so besteht zwischen den drei

Modellen folgende Beziehung:

$$(1 + \frac{\lambda}{2\mu}) \bar{K}_3 \leq \bar{K}_2 < \bar{K}_1.$$

Für $0 < p_1 < \frac{1}{2}$ und $p_2 = \frac{p_1}{1-p_1}$ gilt

$$(1 + \frac{\lambda}{2\mu}) \bar{K}_3 = \bar{K}_2$$

Beweis: Die Behauptung folgt aus folgenden Einzelaussagen:

1. $\bar{K}_2$ ist eine stetige Funktion von p_1 und p_2 für $0 < p_1, p_2 < 1$. $\bar{a} + \bar{b} - 2\bar{a}\bar{b}$
2. Die Fläche $\bar{K}_2 = \frac{\bar{a} + \bar{b} - 2\bar{a}\bar{b}}{(1-\bar{a})(1-\bar{b})}$ hat mit der

$$\text{Ebene } \bar{K}_2 = \frac{2\lambda}{2\mu - \lambda} - \varepsilon$$

- 2.1. für $\varepsilon = 0$ die Punktmenge (p_1', p_2') mit $0 < p_1' < \frac{1}{2}$ und $p_2' = \frac{p_1'}{1-p_1'}$ gemeinsam,
- 2.2. bei $\varepsilon > 0$ keinen Punkt gemeinsam.

3. Die Fläche $\bar{K}_2 = \frac{\bar{a} + \bar{b} - 2\bar{a}\bar{b}}{(1-\bar{a})(1-\bar{b})}$ hat mit der

$$\text{Ebene } K_2 = \frac{\lambda}{\mu - \lambda} + \varepsilon \quad \text{für } \varepsilon \geq 0 \text{ keinen Punkt gemeinsam.}$$

4. Es existiert ein Punkt $(\bar{p}_1, \bar{p}_2)$ mit $0 < \bar{p}_1, \bar{p}_2 < 1$ mit $\frac{2\lambda}{2\mu - \lambda} < \bar{K}_2(\bar{p}_1, \bar{p}_2) < \frac{\lambda}{\mu - \lambda}$

Die Einzelaussagen 1. und 4. sind trivial.

Einzelaussage 2:

Für gemeinsame Punkte der beiden Flächen gilt:

$$\frac{\bar{a} + \bar{b} - 2\bar{a}\bar{b}}{(1-\bar{a})(1-\bar{b})} = \frac{2\lambda}{2\mu - \lambda} - \varepsilon \quad \Leftrightarrow$$

$$\varepsilon (\mu (2\mu^2 - 3\lambda\mu + \lambda^2) (p_1 + p_2 - p_1 p_2)^2 + \lambda^2 (2\mu - \lambda) p_1 (1 - p_1) p_2) + \lambda^2 \mu (p_1 - p_2 (1 - p_1))^2 = 0 \quad (*)$$

Für $\varepsilon = 0$ und $p_1 \neq 1$ folgt $p_2 = \frac{p_1}{1-p_1}$

Da $0 < p_2 < 1$ gelten muß, folgt $0 < p_1 < \frac{1}{2}$.
Also nimmt $\bar{K}_2$ für $0 < p_1 < \frac{1}{2}$ und $p_2' = \frac{p_1'}{1-p_1'}$
den Wert $\frac{2\lambda}{2\mu-\lambda}$ an.

Für $\varepsilon > 0$ sind beide Summanden in (*) positiv,
sofern $\lambda < \mu$ und $0 < p_1, p_2 < 1$ gelten.
Also hat $\bar{K}_2$ mit $\frac{2\lambda}{2\mu-\lambda} - \varepsilon$ für $\varepsilon > 0$ keinen
Punkt gemeinsam.

Einzelaussage 3:

Für gemeinsame Punkte der beiden Flächen gilt

$$\frac{\bar{a} + \bar{b} - 2\bar{a}\bar{b}}{(1-\bar{a})(1-\bar{b})} = \frac{\lambda}{\mu-\lambda} + \varepsilon \quad \Leftrightarrow$$

$$- \lambda^2(\mu-\lambda)p_1(1-p_1)p_2 = \\ \varepsilon(\mu(\mu-\lambda)^2(p_1+p_2-p_1p_2)^2 + \lambda^2(\mu-\lambda)p_1(1-p_1)p_2)$$

Für $0 < p_1, p_2 < 1$ und $\lambda < \mu$ ist die linke Seite
dieser Gleichung negativ, während die rechte
für $\varepsilon = 0$ den Wert 0 hat bzw. für $\varepsilon > 0$ positiv
ist. Also hat $\bar{K}_2$ mit der Ebene $\frac{\lambda}{\mu-\lambda} + \varepsilon$
für $\varepsilon \geq 0$ unter den angegebenen Voraussetzungen
keinen Punkt gemeinsam.

Bild 6 skizziert $\bar{K}_1, \bar{K}_2$ und $\bar{K}_3$ als Funktionen von $\frac{\lambda}{\mu}$.

Bemerkungen: Satz 2.12. kann in zweierlei Hinsicht
als Kriterium herangezogen werden:

1. Der Vergleich der drei Rechenanlagen zeigt, welche
Struktur bei vorgegebenen Programmläufen - jeweils
auf einen speziellen Auftrags - Raum bezogen -
günstig ist.
2. Andererseits kann die Struktur des 2. Modelles als
gegeben angenommen und die Frage gestellt werden,
ob es zweckmäßig erscheint, Programmläufe aus einem
Auftrags - Raum bearbeiten zu lassen. Gegebenenfalls

wird sich dabei zeigen, daß der Rechnerkern sehr stark und das E/A - Steuerwerk nur sehr wenig benutzt werden, also das Modell 2 ein Verhalten zeigt, das dem des 1. Modelles ähnlich ist, wenn dieses dieselben Aufträge auszuführen hätte. In diesem Fall können evtl. Aufträge aus mehreren Auftrags - Räumen so zusammengefaßt werden, daß insgesamt ein günstiges Verhalten resultiert.

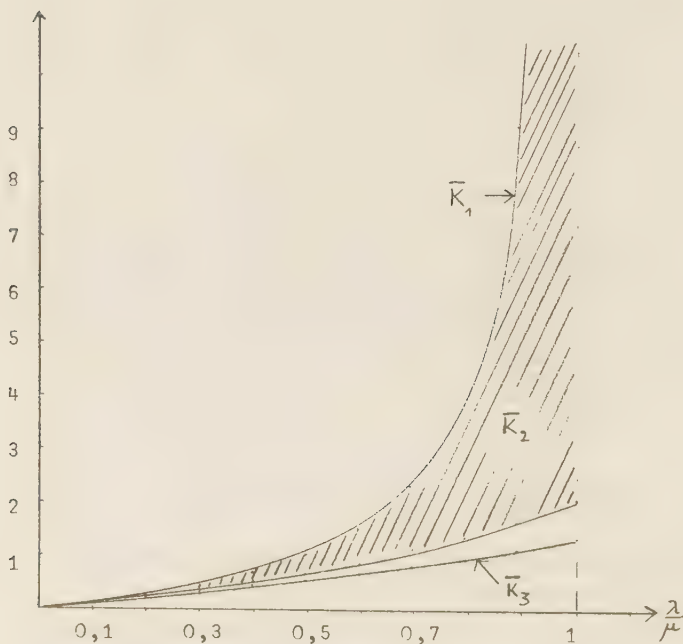


Bild 6: Vergleich zwischen den Modellen
für $\lambda < \mu$.

Beispiel:

Für zwei Auftrags - Räume werden die Mittelwerte von K_1, K_2 und K_3 angegeben. In beiden Fällen sei

$$\lambda = \frac{1}{2000}.$$

Auftrags - Raum 1:

Seien $\mu_1' = \frac{1}{450}$, $\mu_2' = \frac{1}{600}$, $p_1' = \frac{1}{4}$ und $p_2' = \frac{1}{3}$.

Es gelten $\mu_1' p_1' = \mu_2' p_2'$ und $p_2' = \frac{p_1'}{1-p_1'}$

so daß $\bar{K}_2 = \frac{2\lambda}{2\mu - \lambda}$ mit $\mu := \mu_1' p_1'$.

Somit ergeben sich $\bar{K}_1 = 9$, $\bar{K}_2 = 1,64$ und $\bar{K}_3 = 1,13$, also für $\bar{K}_3$, verglichen mit $\bar{K}_1$, ein nur wenig kleinerer Wert als $\bar{K}_2$.

Auftrags - Raum 2:

Seien $\mu_1'' = \frac{1}{1350}$, $\mu_2'' = \frac{1}{900}$, $p_1'' = \frac{3}{4}$ und $p_2'' = \frac{1}{2}$.

Es gelten $\mu_1'' p_1'' = \mu_2'' p_2''$ und $\mu_1' p_1' = \mu_1'' p_1''$, d. h. im Mittel ist die Ausführungsdauer der Programmläufe für Raum 1 und Raum 2 gleich.

Während $\bar{K}_1 = 9$ und $\bar{K}_3 = 1,13$ gelten wie für Raum 1, ergibt sich $\bar{K}_2 = 3,52$.

3. Rechenanlagen mit Speicherhierarchie

3.0. Vorbemerkungen

Eine wesentliche Voraussetzung bei den Untersuchungen des Kapitels 2 besagt, daß die Rechenanlage über einen Primärspeicher von so großer Kapazität verfügen solle, daß alle anstehenden Aufträge in ihm abgelegt werden können. Geht man von dieser Voraussetzung ab, so wird auch die Annahme, Regiewechsel bei Programmläufen seien zu jeder Zeit ohne Zeitverlust möglich, unrealistisch, da u. U. bei einem Regiewechsel Programm und Daten für den zu initiierten Programmlauf zunächst vom Hintergrund - in den Primärspeicher geladen werden müssen. Wäre diese Ladezeit ausschließlich vom Speicherbedarf eines Auftrages abhängig, so könnte sie zur Dauer des entsprechenden Programmlaufes addiert und die Summe als neue Ausführungsdauer angesehen werden; dann könnte - wenigstens für das Modell, das Abschnitt 2.1. zugrunde liegt - das Reihenfolgeproblem mit diesen Werten gelöst werden. Ob ein Umspeicherprozeß vor einem Programmlauf notwendig ist und welche Zeit für seine Ausführung nötig ist, hängt jedoch wesentlich von der aktuellen, also zeitabhängigen Belegung des Primärspeichers und somit von der Lösung des Reihenfolgeproblems ab.

Für die Modellvorstellungen, die in Abschnitt 2.1.1. fixiert wurden, könnte man eine Lösung nach folgendem Ansatz suchen: Gegeben seien $n \in \mathbb{N}$ Aufträge mit der Ausführungsdauer $\tau := (e_1, \dots, e_n)$ und $t_1 := t_2 := \dots := t_n = 0$. Zugelassen seien ausschließlich Ausführungsreihenfolgen, die keine Unterbrechungen zulassen. Für jedes Auftragspaar (i, k) mit $i \neq k$ und $i, k \in \{1, \dots, n\}$ sei in l_{ik} festgelegt, welche Ladezeit nötig ist, wenn Auftrag k unmittelbar nach Auftrag i ausgeführt wird. Berücksichtigen wir also die Matrix $L := (l_{ik})_{nn}$ mit $l_{ii} = 0$ für $\forall i \in \{1, \dots, n\}$ so lautet das entsprechende Reihenfolgeproblem: Gesucht ist eine Permutation von $\{1, 2, \dots, n\}$ derart, daß die mittlere

Wartezeit oder Verweilzeit bei der Ausführung der Aufträge in der Reihenfolge, die durch diese Permutation festgelegt ist, zum Minimum wird. Diese Aufgabe entspricht dem Problem eines Handlungsreisenden, der mit minimalem Aufwand n Städte bereisen soll und ist, wie z. B. in Conway et. al. (1967) gezeigt, entsprechend zu lösen. Ist in diesem Fall $[1], [2], \dots, [n]$ eine optimale Reihenfolge, so ist die Wartezeit des Auftrages $[k]$ mit $k > 1$ gegeben durch $\sum_{i=1}^{k-1} e_{[i]} + \sum_{i=2}^k l_{[i-1][i]}$. Läßt man für den Fall, daß die Ankunftsstermine verschieden sind, Ausführungen mit Unterbrechungen zu, so wird die Beschreibung der notwendigen Ladeprozesse durch eine zeitunabhängige Matrix L i. a. nicht ausreichen, da insbesondere bei jeder Ausführungsunterbrechung ein Ladeprozess nötig sein kann, dessen Dauer von der speziellen Belegung des Primärspeichers im Unterbrechungszeitpunkt abhängig sein wird.

Wir werden im folgenden diesen Lösungsweg nicht gehen. Vielmehr wird gezeigt, wie durch Simulation Reihenfolgeprobleme in Rechenanlagen analysiert werden können. Unsere Bemühungen, Programmläufe in Rechenanlagenmodellen zu analysieren, die realen Anlagen näher stehen, als die Modelle des Kapitels 2, müssen die Tatsache berücksichtigen, daß keine statistischen Angaben über die Dauer von Ein- und Ausgabeoperationen und die Zeitpunkte ihres Auftretens im Verlauf der Gesamtausführung von Aufträgen bekannt sind. In diesem Kapitel werden Programmläufe ohne Ein- und Ausgabeoperationen für Rechenanlagen mit einem Primärspeicher begrenzter Kapazität und einer Speicherhierarchie analysiert. Wir nehmen dabei an, daß alle Eingabeoperationen vor der Ankunft eines Auftrages durchgeführt sind und folglich Programm und Daten bei der Ankunft eines Auftrages im Speicher, und zwar im Tertiärspeicher stehen. Ebenso soll die Ausgabe der Ergebnisse

unabhängig vom Auftrag nach dessen Ausführung erfolgen, und zwar wieder vom Tertiärspeicher zu den Ausgabegegeräten. Diese Annahmen können für Teilnehmer - Rechensysteme zu einer angemessenen Modellvorstellung führen.

Das in Abschnitt 3.1. beschriebene Rechenanlagenmodell berücksichtigt neben der Speicherhierarchie einen Rechnerkern und ein E/A - Steuerwerk. Diese beiden Elemente können gleichzeitig aktiv sein - analog dem Warteschlangenmodell, das in Abschnitt 2.2. analysiert wurde. Allerdings wird das E/A - Steuerwerk ausschließlich zur Ausführung von Umspeicherprozessen benützt. Für dieses Rechenanlagenmodell wird eine geeignete Programmablauf - Beschreibung aus einem gegebenen Auftrags - Raum abgeleitet.

In Abschnitt 3.2. wird ein Warteschlangensystem mit drei Bedienungsstationen - Primärspeicherverwaltung, Rechnerkern und E/A - Steuerwerk - und der Ablauf der Simulation in diesem System beschrieben. Jeder Programmablauf wird dabei in mehrere Phasen (Programmablauf - Teil genannt) aufgeteilt, und jede Phase bezieht sich auf eine der Bedienungsstationen. Entsprechend unterteilen sich die Aufträge, so daß in den Warteschlangen jeweils Teil - Aufträge zur Ausführung anstehen. Die Ausführung der Aufträge wird wieder von einer Strategie des Betriebsprogrammes gesteuert. Strategien werden der Struktur des Warteschlangensystems entsprechend durch drei Abbildungen φ , χ und σ definiert, wobei φ die Primärspeicherzuteilung, χ die Aktivität des E/A - Steuerwerkes und σ die Aktivität des Rechnerkerns steuert.

In Abschnitt 3.3. werden Simulationsergebnisse für folgende Strategien angegeben:

1. Ausführung aller Teil - Aufträge in der Reihenfolge ihrer Ankunft (FIFO).
2. Primärspeicherzuteilung in der Reihenfolge minimalen Speicherbedarfs; Ausführung der Teil - Aufträge durch die übrigen Bedienungsstationen in der Reihenfolge ihrer Ankunft (MSBP).
3. Primärspeicherverwaltung und E/A - Steuerwerk führen die Teil - Aufträge in der Reihenfolge ihrer Ankunft aus; der Rechnerkern fertigt die Teil - Aufträge in der Reihenfolge kürzester Ausführungsdauer ab, jedoch ohne Ausführungsunterbrechungen zuzulassen (KADR).
4. Primärspeicherzuteilung in der Reihenfolge minimalen Speicherbedarfs; Ausführung der Teil - Aufträge durch das E/A - Steuerwerk in der Reihenfolge ihrer Ankunft; Ausführung der Teil - Aufträge durch den Rechnerkern in der Reihenfolge kürzester Ausführungsdauer und ohne Ausführungsunterbrechung (MSPKAP).
5. Ausführung aller Teil - Aufträge in der Reihenfolge kürzester Rechnerkern - Ausführungsdauer einschließlich Ausführungsunterbrechungen und Verdrängungen aus dem Primärspeicher (KADPKR).

Die angegebenen Simulationsergebnisse gelten für ein Modell, das eine Magnettrommel als Sekundärspeicher und Magnetplatten als Tertiärspeicher benutzt.

3.1. Das Modell

Zur Modellbildung gehen wir von der Annahme aus, daß eine Rechenanlage als Teilnehmer - Rechen - system arbeiten soll. Allen Benutzern stehen im Tertiärspeicher der Anlage Privatbibliotheken zur Verfügung, in denen sie über größere Zeiträume hinweg Programme und Daten ablegen können. Die Ein- und Ausgabe erfolge über "langsame" Außenstationen, wie z. B. Fernschreiber. (Zur

Konzeption von Teilnehmer - Rechensystemen siehe z. B. Händler (1968) und Spies (1967)).

Ein Auftrag an das Teilnehmer-Rechensystem liege vor, wenn ein Steuerkommando gegeben ist, das von der Rechenanlage eine Tätigkeit verlangt, die nicht in der Ein- oder Ausgabe einer Zeichenfolge bestehe, also z. B.

Syntaktische Prüfung eines Programmes oder Programmteils,
Übersetzen eines Programmes,
Ausführen eines Programmes.

Zwischen der Eingabe zweier aufeinanderfolgender Aufträge eines Benutzers kann dieser Ein- und Ausgabeoperationen ausführen, die Programme und Daten seiner Privatbibliothek betreffen, oder überlegen, welcher Auftrag als nächster gegeben werden soll. Alle Operationen der Rechenanlage, die dabei ausgeführt werden, sollen jedoch vernachlässigt werden; die Ein- und Ausgabe über die Außenstationen belaste das E/A - Steuerwerk nicht wesentlich. Wir bezeichnen das Zeitintervall zwischen der Eingabe zweier aufeinanderfolgender Aufträge desselben Benutzers als Denkphase, die Länge dieses Intervalles als Denkzeit D . Für die Denkzeit jedes Benutzers sei eine Verteilungsfunktion gegeben, die mit $\mathcal{V}(x)$ bezeichnet wird.

Arbeiten mehrere Benutzer gleichzeitig an dem Teilnehmer-Rechensystem, so wollen wir annehmen, daß die Denkzeit jedes einzelnen Benutzers der Verteilungsfunktion $\mathcal{V}(x)$ genügt.

Die Rechenanlage soll aus folgenden Elementen bestehen:

- ein Rechnerkern
- ein E/A - Steuerwerk
- Primärspeicher der Kapazität M_0
- Sekundärspeicher
- Tertiärspeicher

Dabei hat das E/A - Steuerwerk ausschließlich Umspeicherprozesse zwischen den Hintergrundspeichern und dem Primärspeicher auszuführen.

Für diese Anlage sei ein Auftrags - Raum $(\Omega, \mathcal{P}(\Omega), P)$ gegeben, dem ein Programm - Raum $(\Pi, \mathcal{P}(\Pi), P')$ zugrunde liege. Für alle $\omega \in \Omega$ und alle $t \geq 0$ folgt aus den Annahmen: ist $\omega(t) = (z_1, z_2, z_3, z_4)$, so gilt $z_2 = z_4 = 0$.

Von jedem Auftrag wird im folgenden die virtuelle Ausführungsdauer und der Speicherbedarf benutzt. Die virtuelle Ausführungsdauer ergibt sich wie in Kapitel 2 durch eine Abbildung $E : \Omega \rightarrow \mathbb{R}$. Es wird angenommen, daß für E eine Verteilungsfunktion $\mathcal{E}(x)$ bekannt ist.

Zur Definition des Speicherbedarfs setzen wir voraus, daß zu jedem $\omega \in \Omega$ genau ein $\pi \in \Pi$ mit $\omega \in \Omega_\pi$ existiert, wobei $(\Omega_\pi, \mathcal{P}(\Omega_\pi), P_\pi)$ der Auftrags - Raum bzgl. π sei. Für $\omega_\pi \in \Omega_\pi$ sei

$$M(\omega_\pi) := \max \{ z_3 \mid \exists t (t \in \mathbb{R}^+ \wedge \omega_\pi(t) = (1, 0, z_3, 0)) \}.$$

Dann sei der Speicherbedarf $M(\pi)$ des Programmes π definiert durch $M(\pi) := \max_{\omega_\pi \in \Omega_\pi} \{ M(\omega_\pi) \}$.

Unter den angegebenen Annahmen ist die durch $M(\omega) := M(\pi)$ für alle $\omega \in \Omega_\pi$ definierte Abbildung $M : \Omega \rightarrow \mathbb{R}$ meßbar und M ist eine stochastische Variable. Es wird angenommen, daß für M eine Verteilungsfunktion $\mathcal{M}(x)$ bekannt ist. Zudem seien M und E unabhängig.

Die angegebene Definition des Speicherbedarfs geht davon aus, daß ein Programm einen Speicherbereich fester Größe adressiert und jeder Auftrag den gesamten Speicherbedarf des zugehörigen Programmes anfordert, unabhängig davon, ob der Auftrag diesen Speicherraum während seiner Ausführung benutzt. (Für die Definition des Speicherbedarfs

eines Auftrages, in der die Eigenschaften der Programmläufe berücksichtigt sind, sei auf Kapitel 4 verwiesen.)

Hat ein Benutzer einen Auftrag gegeben, so sollen das entsprechende Programm und die aktuellen Parameter in dessen Privatbibliothek stehen; diese seien für alle Benutzer einheitlich im Tertiärspeicher abgelegt. Entsprechend sollen die Ergebnisse jedes Programmlaufes wieder in die Privatbibliothek übertragen werden. Ergebnisse eines Programmlaufes sind Teile der Information, die bei Abschluß der Auftragsausführung in den Speicherbereichen stehen, in welchen der Programmlauf erfolgte. Sei m der Speicherbedarf eines Auftrages und B die Anzahl der Speicherplätze, die bei Abschluß des Programmlaufes Ergebnisse enthalten. Dann sei B eine stochastische Variable, die den folgenden bedingten Verteilungsfunktionen genüge:

$\mathcal{B}(x|m) :=$ Wahrscheinlichkeit ($B < x|m$) mit

$$\mathcal{B}(x|m) := \begin{cases} 0 & \text{für } x \leq 0 \\ \left\lceil \frac{x}{m} \right\rceil & \text{für } 0 < x \leq m \\ 1 & \text{für } m < x \end{cases}$$

Dabei sei $\left\lceil \frac{x}{m} \right\rceil$ das kleinste ganzzahlige Vielfache von $\frac{1}{m}$, für das $\left\lceil \frac{x}{m} \right\rceil \geq \frac{x}{m}$ gilt. Es wird also angenommen, daß nach Abschluß der Ausführung eines Auftrages, dessen Speicherbedarf m Wörter beträgt, gemäß einer Gleichverteilung $1, 2, \dots, m$ Speicherzellen Ergebnisse enthalten, die in die Privatbibliothek der Benutzer zu übertragen sind.

Nach diesen Vorbereitungen läßt sich das Modell, dessen Verhalten simuliert werden soll, wie folgt beschreiben.

1. An der Rechenanlage, die als Teilnehmer - Rechensystem arbeiten soll, sind NA Benutzer gleichzeitig aktiv; alle Benutzer geben nach einer Denkphase, deren Länge durch $\mathcal{J}(x)$ gegeben ist, Aufträge an die Anlage.

2. Jeder Auftrag ist gekennzeichnet durch drei Angaben:
 - 2.1. Die virtuelle Ausführungsdauer, die durch $\ell(x)$ festgelegt ist; die Ausführung wird vollständig vom Rechnerkern gesteuert.
 - 2.2. Der Speicherbedarf des Auftrages, festgelegt durch $\mathcal{M}(x)$.
 - 2.3. Der Speicherbedarf der Ergebnisse, gegeben durch $\mathcal{L}(x|m)$, wobei m der Speicherbedarf des Auftrages ist.
3. Zum Zeitpunkt der Eingabe jedes Auftrages stehen dessen Programm und Daten im Tertiärspeicher, dort sind auch die Ergebnisse aller Programmläufe abzulegen.

Die Funktion $\mathcal{R}(x)$, $\ell(x)$ und $\mathcal{M}(x)$, welche bei der Durchführung der Simulation benutzt wurden, sind in den Bildern 7, 8 und 9 gezeigt. Sie wurden durch Messungen am System MAC des M.I.T. gewonnen (siehe Scherr (1967)).

Dieser Modellvorstellung entsprechend können wir bzgl. der Ausführung der Aufträge folgende Aussage machen: Der Programmlauf in der realen Rechanlage, der zur Ausführung eines Auftrages notwendig ist, besteht aus wenigstens drei Programmlauf - Teilen:

1. Teil: Umspeicherprozeß zwischen Tertiär- und Primärspeicher; das Programm und die aktuellen Parameter werden aus der Privatbibliothek des Benutzers in den Primärspeicher übertragen, da ausschließlich Information, die in diesem Speicher steht, vom Rechnerkern verarbeitet werden kann.
2. Teil: Ausführung des Auftrages durch den Rechnerkern; während dieser Zeit, die durch die virtuelle Ausführungsdauer des Auftrages festgelegt ist, belegt der Auftrag Primärspeicherraum.
3. Teil: Umspeicherprozeß zwischen Primär- und Tertiärspeicher; die Ergebnisse des Programmlaufes werden in die Privatbibliothek des Benutzers übertragen.

Bei der Ausführung eines Programmlaufes können zusätzliche Umspeicherungen notwendig sein, wenn die Strategie des Betriebsprogrammes zuläßt, daß einem Auftrag, der bereits Teile des Primärspeichers belegt, diese wieder entzogen werden. Diese Maßnahme sei als **V e r - d r ä n g u n g** bezeichnet. In diesen Fällen wird die entsprechende Information in den Sekundärspeicher übertragen und - sobald dem Auftrag wieder Primärspeicherraum zugeteilt ist - von dort ausgelesen. Die in Abschnitt 3.3. angegebene Strategie KADPKR sieht Verdrängungen vor.

Umspeicherprozesse, die zur Ausführung jedes Auftrages notwendig werden, sind Bestandteil der organisatorischen Maßnahmen, die vom Betriebsprogramm zur Koordinierung der Gesamtarbeit des Systems veranlaßt werden. Sie stellen keine Nutzleistung im Sinne eines einzelnen Benutzers dar; dieser hat sein Programm und entsprechende Daten in Operationen, die hier vernachlässigt werden, in den Speicher der Rechenanlage gebracht; mit der Eingabe des entsprechenden Steuerkommandos möchte er veranlassen, daß der Rechnerkern seinen Auftrag ausführt, und zwar genau den Auftrag, dessen Ausführungsablauf in der virtuellen Rechenanlage durch $\omega \in \Omega$ beschrieben ist. Da die Kapazität des Primärspeichers in der realen Rechenanlage nicht ausreicht, die Privatbibliothek aller Benutzer aufzunehmen, werden sie in den Tertiärspeicher verlagert, wodurch die bezeichneten Umspeicherprozesse notwendig werden.

Im Gegensatz zu der virtuellen Ausführungsdauer eines Auftrages, die durch $\{f(x)$ festgelegt ist, wird die Gesamtdauer aller Operationen, die von Rechnerkern und E/A - Steuerwerk bis zum Abschluß der Umspeicherung der Ergebnisse in die Privatbibliothek eines Benutzers ausgeführt werden, als **r e a l e A u s f ü h r u n g s - d a u e r** bezeichnet. Die Differenz zwischen realer und virtueller

Ausführungsdauer ist als Maß für den Koordinierungsaufwand des Betriebsprogrammes anzusehen.

Umspeicherprozesse zwischen verschiedenen Speichermedien werden gemäß spezieller Arbeitsprogramme ausgeführt, die im Systemprogramm der Rechenanlage zur Verfügung stehen. Jedem dieser Arbeitsprogramme entspricht ein Auftrags - Raum. Analog zu der Unterprogrammtechnik, nach der Systemprogramm - Elemente in Benutzerprogramme eingefügt werden, lassen sich auch diese Programm - Läufe in Benutzer - Programm - Läufe einbeziehen. Diese Vorgehensweise soll an dieser Stelle jedoch nicht näher präzisiert werden.

Für die Beschreibung der Umspeicherprozesse werden zwei weitere stochastische Variable eingeführt:

$U_{SE}(m)$ sei die Dauer der Umspeicherung von m Wörtern zwischen Primär- und Sekundärspeicher. Wir nehmen an, daß für $U_{SE}(m)$ bedingte Verteilungsfunktionen $\mathcal{U}_{SE}(x|m)$ für alle Werte von m gegeben sind.

$U_{TE}(m)$ sei die Dauer der Umspeicherung von m Wörtern zwischen Primär- und Tertiärspeicher. Für alle Werte von m seien bedingte Verteilungsfunktionen $\mathcal{U}_{TE}(x|m)$ für $U_{TE}(m)$ gegeben.

Der Verwaltung des Primärspeichers werden folgende Annahmen zugrunde gelegt:

1. Der Primärspeicher ist nach der Methode der Seitenadressierung organisiert. Eine Seite besteht aus 1024 Wörtern. Der Speicherbedarf eines Auftrages, der sich aus $\mathcal{M}(x)$ ergibt, wird auf ganzzahlige Vielfache von 1024 Wörtern ergänzt. Wenn im folgenden der Speicherbedarf M eines Auftrages den Wert m annimmt, so sei diese Modifikation jeweils bereits durchgeführt.
2. Der Speicherbedarf eines Auftrages wird als Einheit in dem Sinne behandelt, daß ein Programmlauf erst beginnen

kann, wenn ihm der Speicherraum, den der entsprechende Auftrag anfordert, vollständig zur Verfügung steht. Für hiervon abweichende Verfahren sei auf Kapitel 4 verwiesen.

Nach diesen Festlegungen ergibt sich als Modell, an dem das Verhalten der Rechenanlage bei der Ausführung von Programmläufen, analysiert werden kann, ein Warteschlangensystem mit folgenden drei Bedienungsstationen: Der Primärspeicher - Verwalter. Vor diesem warten alle Aufträge, denen kein Primärspeicherraum zugeteilt ist; sie befinden sich in der P - Schlange.

Das E/A - Steuerwerk. Vor diesem warten alle Aufträge, für die ein Umspeicherprozeß auszuführen ist. Sie werden in zwei Warteschlangen gesammelt; in der HP - Schlange stehen alle Aufträge, für die eine Umspeicherung aus dem Hintergrund - in den Primärspeicher auszuführen ist; in der PH - Schlange stehen alle Aufträge, für die eine Umspeicherung aus dem Primär- in den Hintergrundspeicher auszuführen ist. Allen Aufträgen, die auf Ausführung einer Umspeicheroperation warten, ist insbesondere der von ihnen angeforderte Primärspeicherraum zuge- teilt.

Der Rechnerkern. Vor diesem warten alle Aufträge, deren Programm und Daten im Primärspeicher stehen, sie befinden sich in der R - Schlange.

Die Ausführung jedes Auftrages besteht in der Ausführung einer Folge von Teil - Aufträgen durch die einzelnen Bedienungsstationen dieses Warteschlangensystems. Diese Folge wird festgelegt durch den Auftrag und durch die Strategie des Betriebsprogrammes, in denen festgelegt ist, nach welchen Kriterien die Bedienungsstationen Teil - Aufträge zur Ausführung auswählen. Bild 10 skizziert das Warteschlangensystem und den Weg der Aufträge durch dieses System für alle Strategien, die Verdrängungen nicht zulassen.

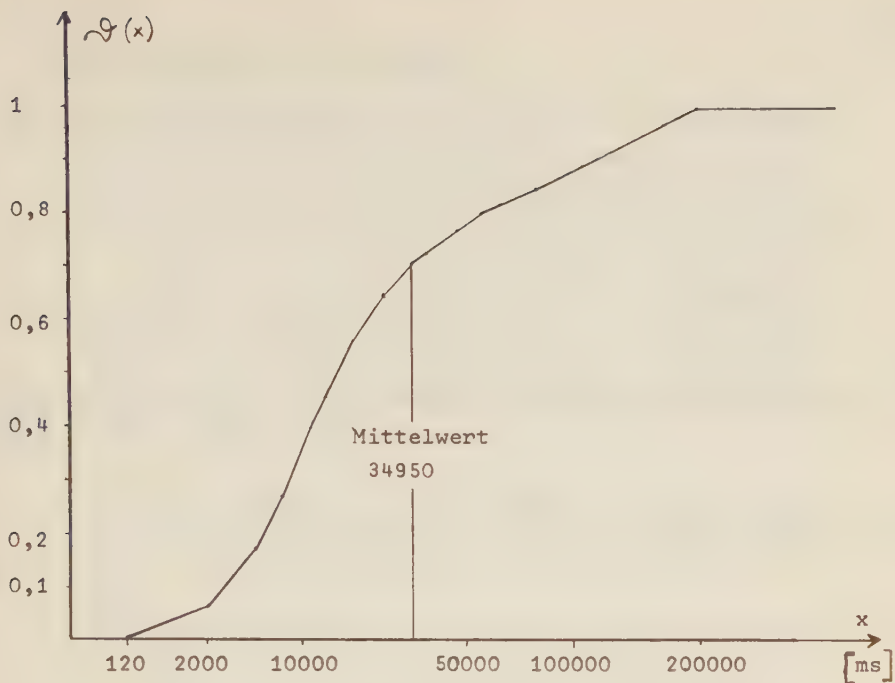


Bild 7: $G(x)$ - Verteilungsfunktion der Denkzeit.

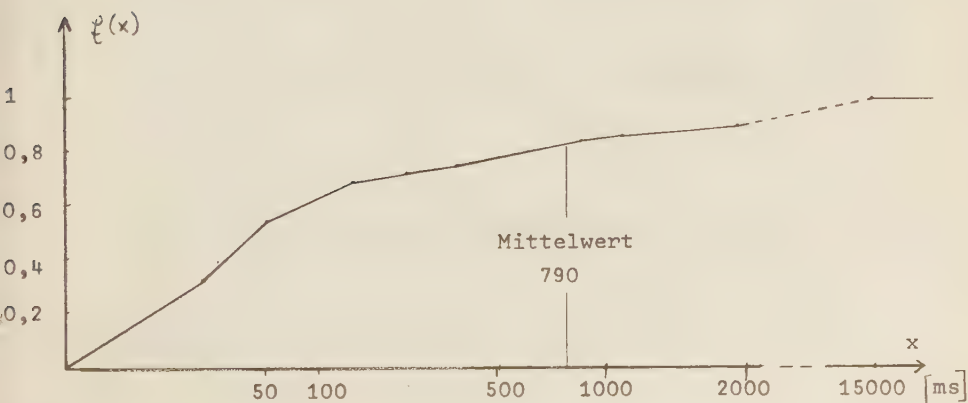


Bild 8: $F(x)$ - Verteilungsfunktion der virtuellen Ausführungs-
dauer.

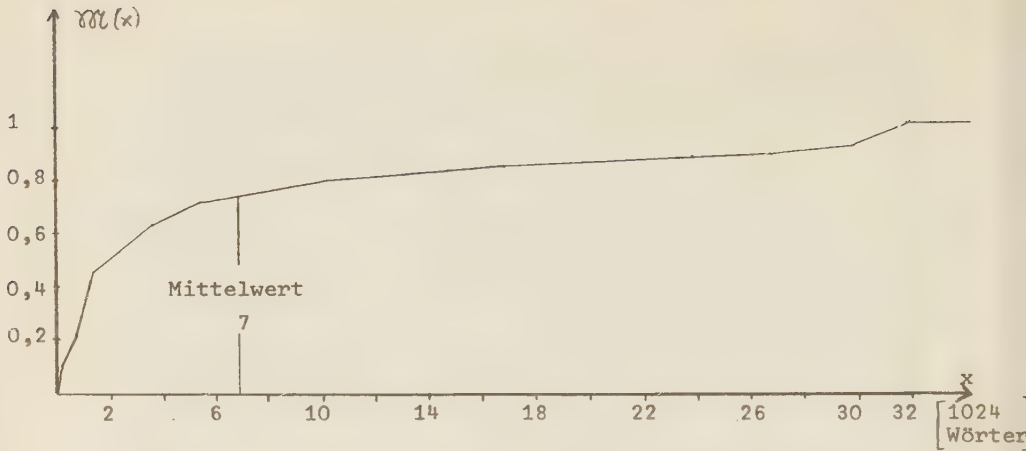


Bild 9: $F(x)$ - Verteilungsfunktion des Speicherbedarfs.

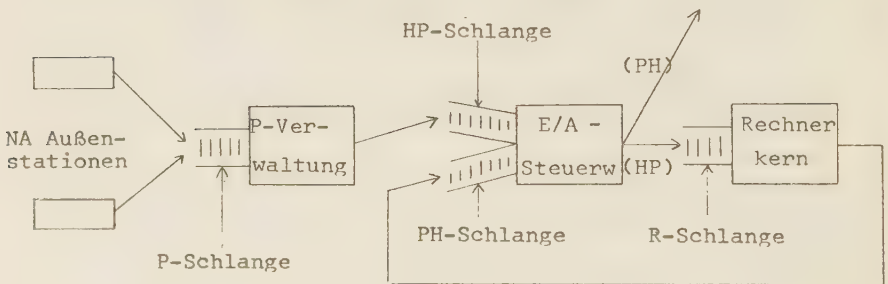


Bild 10: Das Warteschlangensystem

3.2. Der Ablauf der Simulation

Der Simulationsablauf bildet das Verhalten der Benutzer an den Außenstationen und die Ausführung der Aufträge durch die drei Bedienungsstationen des Warteschlangensystems nach. Dem zeitlichen Ablauf wird eine diskrete Zeitskala zugrunde gelegt. Die Simulation beginne im Zeitpunkt $t = 0$. Der diskreten Zeitskala entsprechend nehmen wir an, daß alle stochastische Variable ausschließlich Werte aus einer endlichen Untermenge der natürlichen Zahlen mit positiver Wahrscheinlichkeit annehmen.

Für jeden Zeitpunkt wird das Gesamtsystem durch einen Zustand charakterisiert, so daß der Simulationsablauf als eine Folge von Zuständen beschrieben werden kann.

Nach Konstruktion ist der Zustand eines Benutzers zu einem beliebigen Zeitpunkt t vollständig beschrieben durch die Länge seiner Denkphase, also durch Angabe der Zeit, die bis zur Eingabe seines nächsten Auftrages vergehen wird; sie sei $d(t)$. Ist $d(t) = 0$, so gibt der Benutzer zum Zeitpunkt t einen Auftrag an das System und beginnt eine neue Denkphase.

Der Zustand des Warteschlangensystems zu einem Zeitpunkt t läßt sich vollständig beschreiben durch die Aufträge, die sich zu diesem Zeitpunkt im System befinden und durch deren Betriebsmittelanforderungen. Dies erfolgt in einer Matrix Q , die in vier Teil-Matrizen unterteilt ist. Dabei beschreibt Q^1 den Zustand der P-Schlange, Q^2 den Zustand der HP-Schlange, Q^3 die PH-Schlange und Q^4 die R-Schlange. Ist $q_j^i(t)$ die j -te Zeile der Teil-Matrix Q^i , so beschreibt diese einen Auftrag, der in der i -ten Warteschlange ansteht.

Die Strategie des Betriebsprogrammes ist festgelegt durch drei Abbildungen: σ legt in Abhängigkeit von $Q^4(t)$ fest, welcher der anstehenden Aufträge im Zeitintervall $[t, t+1)$ vom Rechnerkern ausgeführt wird. χ legt in Abhängigkeit von $Q^2(t)$ und $Q^3(t)$ fest, welcher der anstehenden Aufträge im Zeitintervall $[t, t+1)$ vom E/A - Steuerwerk ausgeführt wird. ρ legt in Abhängigkeit von $Q(t)$ fest, wie der verfügbare Primärspeicherraum zum Zeitpunkt t den anstehenden Aufträgen zugeteilt wird.

Nach diesen Vorbemerkungen wird der Simulationsablauf im folgenden exakt beschrieben.

Def. 3.1. Ist $NA \in \mathbb{N}$ die Anzahl der aktiven Benutzer und D' der maximale Wert, den D mit positiver Wahrscheinlichkeit annimmt. Dann sei $\mathcal{D}(t) := (d_1(t), \dots, d_{NA}(t))$ mit $\mathcal{D}(t) \in \mathbb{Z}_{D'}^{NA}$ der Zustand der NA aktiven Benutzer zum Zeitpunkt t .

Def. 3.2. Ist $\mathcal{L}(x)$ Verteilungsfunktion der stochastischen Variablen C , so bedeute $c := [\mathcal{L}(x)]$: c wird ein Wert aus dem Wertevorrat von C , gewürfelt gemäß $\mathcal{L}(x)$ zugewiesen.

Def. 3.3. Sei M' der maximale Wert, den M mit positiver Wahrscheinlichkeit annimmt, E' der entsprechende Wert für E , U' der maximale Wert, den $U_{SE}(m)$ bzw. $U_{TE}(m)$ für $m \in \{1, \dots, M'\}$ annehmen, und $H' := \max\{E', U'\}$. Dann sei $q(t) := (q_1(t), q_2(t), \dots, q_6(t))$ mit $q(t) \in \mathbb{Z}_{M'}^3 \times \mathbb{Z}_{E'} \times \mathbb{Z}_{U'} \times \mathbb{Z}_{H'}$ der Zustand eines Auftrages. Dabei bedeuten:

- $q_1(t)$ die Anzahl der Seiten, die dem Auftrag zum Zeitpunkt t im Primärspeicher zugeteilt sind;
- $q_2(t)$ die Anzahl der Seiten, die dem Auftrag zum Zeitpunkt t im Sekundärspeicher zugeteilt sind;
- $q_3(t)$ die Anzahl der Seiten, die dem Auftrag zum Zeitpunkt t im Tertiärspeicher zugeteilt sind;
- $q_4(t)$ die aktuelle Rechnerkern - Ausführungsdauer des Auftrages zum Zeitpunkt t ;

$q_5(t)$ die aktuelle Ausführungsdauer eines Umspeicherprozesses bzgl. dieses Auftrages zum Zeitpunkt t ;

$q_6(t)$ die Gesamtausführungsdauer des Teil - Auftrages, der zum Zeitpunkt t in Ausführung ist, bzw. auf dessen Ausführung zum Zeitpunkt t gewartet wird.

Def. 3.4. Sei L die maximale Länge einer Warteschlange.

$Q^1(t) := \begin{pmatrix} q_1^1(t) \\ \vdots \\ q_L^1(t) \end{pmatrix}$ sei der Zustand der P - Schlange;

$Q^2(t) := \begin{pmatrix} q_1^2(t) \\ \vdots \\ q_L^2(t) \end{pmatrix}$ sei der Zustand der HP - Schlange;

$Q^3(t) := \begin{pmatrix} q_1^3(t) \\ \vdots \\ q_L^3(t) \end{pmatrix}$ sei der Zustand der PH - Schlange;

$Q^4(t) := \begin{pmatrix} q_1^4(t) \\ \vdots \\ q_L^4(t) \end{pmatrix}$ sei der Zustand der R - Schlange, wobei $q_j^i(t) \in \mathbb{Z}_{H'}^{\times} \mathbb{Z}_{E'}^{\times} \mathbb{Z}_{U'}^{\times} \mathbb{Z}_{H''}^{\times}$ gilt für $i \in \{1, \dots, 4\}$ und $j \in \{1, \dots, L\}$.

$Q(t) := \begin{pmatrix} Q^1(t) \\ Q^2(t) \\ Q^3(t) \\ Q^4(t) \end{pmatrix}$ sei der Zustand

des Warteschlangensystems zum Zeitpunkt t .

Def. 3.5. Für $l \in \mathbb{N}$ sei $X(l)$ die Menge der l - zeiligen Matrizen

$\begin{pmatrix} p_1 \\ \vdots \\ p_l \end{pmatrix}$ mit $p_i \in \mathbb{Z}_{H'}^{\times} \mathbb{Z}_{E'}^{\times} \mathbb{Z}_{U'}^{\times} \mathbb{Z}_{H''}^{\times}$ für $i \in \{1, \dots, l\}$.

Def. 3.6. Die Strategie σ des Rechnerkerns ist eine Abbildung

$$\sigma: \mathcal{H}(L) \rightarrow \mathbb{Z}_L$$

σ wählt aus der Menge der Aufträge, die in der R - Schlange auf Ausführung warten, keinen oder genau einen Auftrag aus. Das Bild von $Q^4(t)$ bzgl. σ ist entweder 0 oder der Zeilenindex bzgl. Q^4 des Teil - Auftrages, der im Zeitintervall $[t, t+1)$ ausgeführt wird.

Def. 3.7. Die Strategie χ des E/A - Steuerwerkes ist eine Abbildung

$$\chi: \mathcal{H}(2L) \rightarrow \mathbb{Z}_{2L}$$

χ wählt aus der Menge der Aufträge, die in der HP - bzw. PH - Schlange auf Ausführung warten, keinen oder genau einen Auftrag aus. Das Bild von $\begin{pmatrix} Q^2(t) \\ Q^3(t) \end{pmatrix}$ bzgl. χ ist 0, oder der Zeilenindex bzgl. Q^2 oder $L + \text{Zeilenindex bzgl. } Q^3$ des Teil - Auftrages, der im Zeitintervall $[t, t+1)$ ausgeführt wird.

Def. 3.8. Die Strategie ρ der Primär - speicher - Verwaltung ist eine Abbildung $\rho: \mathcal{H}(4L) \rightarrow \mathcal{H}(4L)$.

Im allgemeinen Fall kann die Primärspeicher-Verwaltung Veränderungen in allen Warteschlangen des Systems durchführen.

Def. 3.9. Eine Strategie S des Betriebsprogrammes ist ein Tripel $S := (\rho, \chi, \sigma)$, wobei ρ eine Strategie der Primärspeicher - Verwaltung, χ eine Strategie des E/A - Steuerwerkes und σ eine Strategie des Rechnerkerns sind.

Def. 3.10. Der Zustand des Gesamtsystems zum Zeitpunkt $t = 0$ ist festgelegt durch:

1. $d_n(0) := [\neg \mathcal{P}(x)]$ für $\forall n \in \{1, \dots, NA\}$
2. $Q(0) := 0$

D. h. im Zeitpunkt $t = 0$ befinde sich kein Auftrag im System; alle NA Benutzer beginnen mit einer Denkphase.

Def. 3.11. Der Zustand des Gesamtsystems zum Zeitpunkt $t + 1$ ist festgelegt durch $\mathcal{P}(t)$, $Q(t)$, g , x und σ .

Im einzelnen gelten:

1. Für die Denkphasen:

$$d_n(t+1) := \begin{cases} d_n(t) - 1 & \text{falls } d_n(t) > 1 \\ [\neg \mathcal{P}(x)] & \text{falls } d_n(t) = 1 \end{cases}$$

für alle $n \in \{1, \dots, NA\}$

$$\text{Sei } I_d(t+1) := \{ n | n \in \{1, \dots, NA\} \wedge d_n(t) = 1 \}$$

2. Für das Warteschlangensystem:

- 2.1. Aktivität des Rechnerkerns:

Ist $\sigma(Q^4(t)) = 0$, so gelten

$$1. \mathcal{R}(t+1) := (r_1(t+1), r_2(t+1), \dots, r_6(t+1)) := 0$$

$$2. \bar{Q}^4(t+1) := Q^4(t)$$

Ist $\sigma(Q^4(t)) = i$ mit $i \neq 0$ und $q_{i4}^4(t) > 1$, so gelten:

$$1. \mathcal{R}(t+1) := 0$$

$$2. \bar{Q}^4(t+1) := (\bar{q}_{1m}^4(t+1))_{L6} \text{ mit}$$

$$\bar{q}_{i4}^4(t+1) := q_{i4}^4(t) - 1$$

$$\bar{q}_{i2}^4(t+1) := 0$$

$$\bar{q}_{im}^4(t+1) := q_{im}^4(t) \text{ für } \forall m \in \{1, \dots, 6\} - \{2, 4\}$$

$$\bar{q}_1^4(t+1) := q_1^4(t) \text{ für } \forall l \in \{1, \dots, L\} - \{i\}$$

Ist $\phi(Q^4(t))=i$ mit $i \neq 0$ und $q_{i4}^4(t) = 1$,

so gelten:

1. $r(t+1):=(r_1(t+1),\dots,r_6(t+1))$ mit

$$r_1(t+1):=r_3(t+1):=[f_r(x|q_{i1}^4(t))]$$

$$r_2(t+1):=r_4(t+1) := 0$$

$$r_5(t+1):=r_6(t+1) := [\mathcal{U}_{TE}(x|r_1(t+1))]$$

2. $\bar{Q}^4(t+1) := (\bar{q}_1^4(t+1))_L$ mit

$$\bar{q}_1^4(t+1) := q_1^4(t) \text{ für } \forall l \in \{1, \dots, i-1\}$$

$$\bar{q}_l^4(t+1) := q_{l+1}^4(t) \text{ für } \forall l \in \{i, \dots, L-1\}$$

$$\bar{q}_L^4(t+1) := 0$$

2.2. Aktivität des E/A - Steuerwerkes:

Ist $\chi \begin{pmatrix} Q^2(t) \\ Q^3(t) \end{pmatrix} = 0$, so gelten:

1. $\mathcal{P}(t+1):=(k_1(t+1),\dots,k_6(t+1)) := 0$

2. $\bar{Q}^2(t+1) := Q^2(t)$

3. $\bar{Q}^3(t+1) := Q^3(t)$

Ist $\chi \begin{pmatrix} Q^2(t) \\ Q^3(t) \end{pmatrix} = i$ mit $i \in \{1, \dots, L\}$

und $q_{i5}^2(t) > 1$, so gelten:

1. $\mathcal{P}(t+1) := 0$

2. $\bar{Q}^2(t+1) := (\bar{q}_{lm}^2(t+1))_{L6}$ mit

$$\bar{q}_{i5}^2(t+1) := q_{i5}^2(t) - 1$$

$$\bar{q}_{lm}^2(t+1) := q_{lm}^2(t) \text{ für } \forall m \in \{1, \dots, 6\} - \{5\}$$

$$\bar{q}_l^2(t+1) := q_l^2(t) \text{ für } \forall l \in \{1, \dots, L\} - \{i\}$$

3. $\bar{Q}^3(t+1) := Q^3(t)$

Ist $x \begin{pmatrix} Q^2(t) \\ Q^3(t) \end{pmatrix} = i$ mit $i \in \{1, \dots, L\}$ und $q_{i5}^2(t) = 1$, so gelten:

1. $\mathcal{P}(t+1) := (k_1(t+1), \dots, k_6(t+1))$ mit

$$k_1(t+1) := q_{i1}^2(t)$$

$$k_2(t+1) := q_{i2}^2(t)$$

$$k_3(t+1) := k_5(t+1) := 0$$

$$k_4(t+1) := k_6(t+1) := q_{i4}^2(t)$$
2. $\bar{Q}^2(t+1) := (\bar{q}_1^2(t+1))_L$ mit

$$\bar{q}_1^2(t+1) := q_1^2(t) \text{ für } \forall l \in \{1, \dots, i-1\}$$

$$\bar{q}_1^2(t+1) := q_{l+1}^2(t) \text{ für } \forall l \in \{i, \dots, L-1\}$$

$$\bar{q}_L^2(t+1) := 0$$
3. $\bar{Q}^3(t+1) := Q^3(t)$

Ist $x \begin{pmatrix} Q^2(t) \\ Q^3(t) \end{pmatrix} = L + i$ mit $i \in \{1, \dots, L\}$ und $q_{i5}^3(t) > 1$, so gelten:

1. $\mathcal{P}(t+1) := 0$
2. $\bar{Q}^2(t+1) := Q^2(t)$
3. $\bar{Q}^3(t+1) := (\bar{q}_{1m}^3(t+1))_{L6}$ mit

$$\bar{q}_{i5}^3(t+1) := q_{i5}^3(t) - 1$$

$$\bar{q}_{im}^3(t+1) := q_{im}^3(t) \text{ für } \forall m \in \{1, \dots, 6\} - \{5\}$$

$$\bar{q}_1^3(t+1) := q_1^3(t) \text{ für } \forall l \in \{1, \dots, L\} - \{i\}$$

Ist $x \begin{pmatrix} Q^2(t) \\ Q^3(t) \end{pmatrix} = L + i$ mit $i \in \{1, \dots, L\}$ und $q_{i5}^3(t) = 1$, so gelten:

1. $\mathcal{P}(t+1) := (k_1(t+1), \dots, k_6(t+1))$ mit

$$\mathcal{P}(t+1) := 0, \text{ falls } q_{i4}^3(t) = 0 \text{ bzw.}$$

$$k_1(t+1) := k_5(t+1) := k_6(t+1) := 0$$

$$k_2(t+1) := q_{i2}^3(t)$$

$$k_3(t+1) := q_{i3}^3(t)$$

$$k_4(t+1) := q_{i4}^3(t), \text{ falls } q_{i4}^3(t) \neq 0$$

$$2. \bar{Q}^2(t+1) := Q^2(t)$$

$$3. \bar{Q}^3(t+1) := (\bar{q}_1^3(t+1))_L \text{ mit}$$

$$\bar{q}_1^3(t+1) := q_1^3(t) \text{ für } \forall l \in \{1, \dots, i-1\}$$

$$\bar{q}_1^3(t+1) := q_{l+1}^3(t) \text{ für } \forall l \in \{i, \dots, L-1\}$$

$$\bar{q}_L^3(t+1) := 0$$

2.3. Eingabe von Aufträgen

Ein Benutzer - Auftrag wird genau dann zum Zeitpunkt $t+1$ eingegeben, wenn $I_d(t+1) \neq \emptyset$ ist. Diese Aufträge gelangen alle zunächst in die P - Schlange.

Ist $I_d(t+1) = \emptyset$, so gilt $\bar{Q}^1(t+1) := Q^1(t)$

Ist $I_d(t+1) \neq \emptyset$, so sei $i := |I_d(t+1)|$ und

$$j := \min\{l | l \in \{1, \dots, L\} \wedge q_l^1(t) = 0\}$$

L sei so gewählt, daß j immer existiert und $i+j \leq L+1$. Dann gelten:

$$\bar{Q}^1(t+1) := (\bar{q}_{lm}^1(t+1))_{L6} \text{ mit}$$

$$\bar{q}_1^1(t+1) := q_1^1(t) \text{ für } \forall l \in \{1, \dots, j-1\}$$

$$\bar{q}_{11}^1(t+1) := \bar{q}_{12}^1(t+1) := \bar{q}_{15}^1(t+1) := \bar{q}_{16}^1(t+1) := 0$$

$$\text{für } \forall l \in \{j, \dots, j+i-1\}$$

$$\bar{q}_{13}^1(t+1) := [\mathcal{M}(x)] \text{ für } \forall l \in \{j, \dots, j+i-1\}$$

$$\bar{q}_{14}^1(t+1) := [\mathcal{F}(x)] \text{ für } \forall l \in \{j, \dots, j+i-1\}$$

$$\bar{q}_1^1(t+1) := 0 \text{ für } \forall l \in \{i+j, \dots, L\}$$

2.4. Reaktion auf den Ausführungsabschluß von Teil - Aufträgen.

Sind $w(t+1) = 0$ und $\bar{p}(t+1) = 0$, so gelten:

$$1. \tilde{Q}^1(t+1) := \bar{Q}^1(t+1)$$

$$2. \tilde{Q}^2(t+1) := \bar{Q}^2(t+1)$$

$$3. \tilde{Q}^3(t+1) := \bar{Q}^3(t+1)$$

$$4. \tilde{Q}^4(t+1) := \bar{Q}^4(t+1)$$

Sind $w(t+1) = 0$ und $k_1(t+1) \neq 0$, so gelten:

$$1. \tilde{Q}^1(t+1) := \bar{Q}^1(t+1)$$

$$2. \tilde{Q}^2(t+1) := \bar{Q}^2(t+1)$$

$$3. \tilde{Q}^3(t+1) := \bar{Q}^3(t+1)$$

$$4. \text{Sei } i := \min \{ l | l \in \{1, \dots, L\} \wedge \bar{q}_l^4(t+1) = 0 \}$$

L sei so gewählt, daß i immer existiert.

Dann gelten:

$$\tilde{Q}^4(t+1) := (\tilde{q}_l^4(t+1))_L \text{ mit}$$

$$\tilde{q}_l^4(t+1) := \bar{q}_l^4(t+1) \text{ für } \forall l \in \{1, \dots, L\} - \{i\}$$

$$\tilde{q}_i^4(t+1) := \bar{p}(t+1)$$

Sind $w(t+1) = 0$, $k_1(t+1) = 0$ und $\bar{p}(t+1) \neq 0$, so gelten:

$$1. \text{Sei } i := \min \{ l | l \in \{1, \dots, L\} \wedge \bar{q}_l^1(t+1) = 0 \}$$

L sei so gewählt, daß i immer existiert.

Dann gelten:

$$\tilde{Q}^1(t+1) := (\tilde{q}_l^1(t+1))_L \text{ mit}$$

$$\tilde{q}_l^1(t+1) := \bar{q}_l^1(t+1) \text{ für } \forall l \in \{1, \dots, L\} - \{i\}$$

$$\tilde{q}_i^1(t+1) := \bar{p}(t+1)$$

$$2. \tilde{Q}^2(t+1) := \bar{Q}^2(t+1)$$

$$3. \tilde{Q}^3(t+1) := \bar{Q}^3(t+1)$$

$$4. \tilde{Q}^4(t+1) := \bar{Q}^4(t+1)$$

ind $\nu(t+1) \neq 0$ und $\varphi(t+1) = 0$, so gelten:

$$1. \tilde{Q}^1(t+1) := \bar{Q}^1(t+1)$$

$$2. \tilde{Q}^2(t+1) := \bar{Q}^2(t+1)$$

$$3. \text{Sei } i := \min \{ l \mid l \in \{1, \dots, L\} \wedge \bar{q}_l^3(t+1) = 0 \}$$

L sei so gewählt, daß i immer existiert.

Dann gelten:

$$\tilde{Q}^3(t+1) := (\tilde{q}_l^3(t+1))_L \text{ mit}$$

$$\tilde{q}_l^3(t+1) := \bar{q}_l^3(t+1) \text{ für } \forall l \in \{1, \dots, L\} - \{i\}$$

$$\tilde{q}_i^3(t+1) := \nu(t+1)$$

$$4. \tilde{Q}^4(t+1) := \bar{Q}^4(t+1)$$

Sind $\nu(t+1) \neq 0$ und $k_1(t+1) \neq 0$, so gelten:

$$1. \tilde{Q}^1(t+1) := \bar{Q}^1(t+1)$$

$$2. \tilde{Q}^2(t+1) := \bar{Q}^2(t+1)$$

$$3. \text{Sei } i := \min \{ l \mid l \in \{1, \dots, L\} \wedge \bar{q}_l^3(t+1) = 0 \}$$

L sei so gewählt, daß i immer existiert.

Dann gelten:

$$\tilde{Q}^3(t+1) := (\tilde{q}_l^3(t+1))_L \text{ mit}$$

$$\tilde{q}_l^3(t+1) := \bar{q}_l^3(t+1) \text{ für } \forall l \in \{1, \dots, L\} - \{i\}$$

$$\tilde{q}_i^3(t+1) := \nu(t+1)$$

$$4. \text{Sei } j := \min \{ l \mid l \in \{1, \dots, L\} \wedge \bar{q}_l^4(t+1) = 0 \}$$

L sei so gewählt, daß j immer existiert.

Dann gelten:

$$\tilde{Q}^4(t+1) := (\tilde{q}_l^4(t+1))_L \text{ mit}$$

$$\tilde{q}_l^4(t+1) := \bar{q}_l^4(t+1) \text{ für } \forall l \in \{1, \dots, L\} - \{j\}$$

$$\tilde{q}_j^4(t+1) := \varphi(t+1)$$

Sind $w(t+1) \neq 0$, $k_1(t+1) = 0$ und $\varnothing(t+1) \neq 0$,
so gelten:

1. Sei $i := \min \{ l | l \in \{1, \dots, L\} \wedge \bar{q}_l^1(t+1) = 0 \}$

L sei so gewählt, daß i immer existiert.

Dann gelten:

$$\tilde{Q}^1(t+1) := (\bar{q}_l^1(t+1))_L \text{ mit}$$

$$\bar{q}_l^1(t+1) := \bar{q}_l^1(t+1) \text{ für } \forall l \in \{1, \dots, L\} - \{i\}$$

$$\bar{q}_i^1(t+1) := \varnothing(t+1)$$

$$2. \tilde{Q}^2(t+1) := \bar{Q}^2(t+1)$$

3. Sei $j := \min \{ l | l \in \{1, \dots, L\} \wedge \bar{q}_l^3(t+1) = 0 \}$

L sei so gewählt, daß j immer existiert.

Dann gelten:

$$\tilde{Q}^3(t+1) := (\bar{q}_l^3(t+1))_L \text{ mit}$$

$$\bar{q}_l^3(t+1) := \bar{q}_l^3(t+1) \text{ für } \forall l \in \{1, \dots, L\} - \{j\}$$

$$\bar{q}_j^3(t+1) := w(t+1)$$

$$4. \tilde{Q}^4(t+1) := \bar{Q}^4(t+1).$$

2.5. Primärspeicherzuweisung

Die Primärspeicherbelegung zum Zeitpunkt $t+1$
ergibt sich als Bild von $\tilde{Q}(t+1)$ bzgl. φ .

Damit ist der Zustand des Systems zum Zeitpunkt
 $t+1$:

$$Q(t+1) := \varphi(\tilde{Q}(t+1))$$

Nachdem in den Definitionen 3.1. bis 3.11. der Simulations-
ablauf festgelegt wurde, werden im folgenden die Kenndaten
des simulierten Systems definiert, soweit sie für die Ana-
lyse der Rechenanlage, die in Abschnitt 3.3. am Beispiel
durchgeführt wird, notwendig sind. Es sei jedoch darauf
hingewiesen, daß die Zustandsfolgen, die sich aus den Defi-
nitionen 3.10. und 3.11. ergeben, detaillierte Analysen
der Ausführung jedes Teil - Auftrages gestatten, wobei auch

Aspekte berücksichtigt werden können, die hier nicht genannt sind.

Sei T die Dauer einer Simulation und $t \in [0, T]$

Def. 3.12. Der Rechnerkern ist im Zeitintervall $[t, t+1)$

$$\text{aktiv} : \Leftrightarrow \delta(Q^4(t)) \neq 0.$$

Der Rechnerkern wartet im Zeitintervall

$$: \Leftrightarrow \delta(Q^4(t)) = 0$$

Def. 3.13. Das E/A - Steuerwerk ist im Zeitintervall $[t, t+1)$

$$\text{aktiv} : \Leftrightarrow x \begin{pmatrix} Q^2(t) \\ Q^3(t) \end{pmatrix} \neq 0$$

Das E/A - Steuerwerk wartet im Zeitintervall

$$[t, t+1) : \Leftrightarrow x \begin{pmatrix} Q^2(t) \\ Q^3(t) \end{pmatrix} = 0$$

Def. 3.14. Sei $a_{RK}(T)$ die Anzahl der Zeitintervalle während des Simulationsablaufes, in denen der Rechnerkern aktiv ist, also

$$a_{RK}(0) := 0 \quad \text{und für } t \in [1, T]$$

$$a_{RK}(t) := \begin{cases} a_{RK}(t-1), & \text{falls } \delta(Q^4(t-1)) = 0 \\ a_{RK}(t-1)+1, & \text{falls } \delta(Q^4(t-1)) \neq 0 \end{cases}$$

Dann ist die mittlere Auslastung $\bar{a}_{RK}(T)$ des Rechnerkerns im Zeitintervall $[0, T]$:

$$\bar{a}_{RK}(T) := \frac{a_{RK}(T)}{T}$$

Def. 3.15. Sei $a_K(T)$ die Anzahl der Zeitintervalle während des Simulationsablaufes, in denen das E/A - Steuerwerk aktiv ist, also

$$a_K(0) := 0 \quad \text{und für } t \in [1, T]$$

$$a_K(t) := \begin{cases} a_K(t-1), & \text{falls } x \begin{pmatrix} Q^2(t-1) \\ Q^3(t-1) \end{pmatrix} = 0 \\ a_K(t-1)+1, & \text{falls } x \begin{pmatrix} Q^2(t-1) \\ Q^3(t-1) \end{pmatrix} \neq 0 \end{cases}$$

Dann ist die mittlere Auslastung $\bar{a}_K(T)$ des E/A - Steuerwerkes im Zeitintervall $[0, T]$:

$$\bar{a}_K(T) := \frac{a_K(T)}{T}$$

Def. 3.16. Sei $n(T)$ die Anzahl der während des Simulationsablaufes vollständig ausgeführten Aufträge, also $n(0) := 0$ und für $t \in [1, T]$

$$n(t) := \begin{cases} n(t-1)+1, & \text{falls } x \begin{pmatrix} Q^2(t-1) \\ Q^3(t-1) \end{pmatrix} = L + i \\ & \text{mit } i \in \{1, \dots, L\} \text{ und} \\ & q_{i5}^3(t) = 1 \text{ und } q_{i4}^3(t) = 0 \\ n(t-1) & \text{sonst} \end{cases}$$

Im folgenden wird T so groß angenommen, daß $n(T) > 0$ gilt. Dann ist die mittlere reale Ausführungsdauer $\bar{r}(T)$ eines Auftrages im Zeitintervall $[0, T]$:

$$\bar{r}(T) := \frac{a_{RK}(T) + a_K(T)}{n(T)}$$

Def. 3.17. Sei $L^i(t)$ die Anzahl der Zeilen von $Q^i(t)$, die nicht den Nullvektor enthalten für $i \in \{1, \dots, 4\}$; $L^i(t)$ ist die Länge der Warteschlange i zum Zeitpunkt t .

Dann ist die mittlere Verweilzeit $\bar{v}(T)$ eines Auftrages für das Zeitintervall $[0, T]$:

$$\bar{v}(T) := \left(\sum_{t=0}^T \sum_{i=1}^4 L^i(t) \right) / n(T)$$

Def. 3.18. Die mittlere Wartezeit $\bar{w}_P(T)$ eines Auftrages auf P-Zuteilung im Zeitintervall $[0, T]$ sei:

$$\bar{w}_P(T) := \left(\sum_{t=0}^T L^1(t) \right) / n(T)$$

Def. 3.19. Sei $\bar{L}^2(t)$ die Anzahl der Aufträge, die im Zeitintervall $[t, t+1)$ auf HP - Umspeicherung warten, also

$$\bar{L}^2(t) := \begin{cases} L^2(t)-1, & \text{falls } x \binom{Q^2(t)}{Q^3(t)} \in \{1, \dots, L\} \\ L^2(t) & \text{sonst} \end{cases}$$

Dann ist die mittlere Wartezeit $\bar{w}_{HP}(T)$ eines Auftrages auf HP - Umspeicherung für das Zeitintervall $[0, T]$:

$$\bar{w}_{HP}(T) := \left(\sum_{t=0}^T \bar{L}^2(t) \right) / n(T)$$

Def. 3.20. Sei $\bar{L}^3(t)$ die Anzahl der Aufträge, die im Zeitintervall $[t, t+1)$ auf PH - Umspeicherung warten, also

$$\bar{L}^3(t) := \begin{cases} L^3(t)-1, & \text{falls } x \binom{Q^2(t)}{Q^3(t)} \in \{L+1, \dots, 2L\} \\ L^3(t) & \text{sonst} \end{cases}$$

Dann ist die mittlere Wartezeit $\bar{w}_{PH}(T)$ eines Auftrages auf PH - Umspeicherung für das Zeitintervall $[0, T]$:

$$\bar{w}_{PH}(T) := \left(\sum_{t=0}^T \bar{L}^3(t) \right) / n(T)$$

Def. 3.21. Sei $\bar{L}^4(t)$ die Anzahl der Aufträge, die im Zeitintervall $[t, t+1)$ auf Ausführung durch den Rechnerkern warten, also

$$\bar{L}^4(t) := \begin{cases} L^4(t)-1, & \text{falls } \sigma(Q^4(t)) \neq 0 \\ L^4(t) & \text{sonst} \end{cases}$$

Dann ist die mittlere Wartezeit $\bar{w}_{RK}(T)$ eines Auftrages auf RK - Ausführung für das Zeitintervall $[0, T]$:

$$\bar{w}_{RK}(T) := \left(\sum_{t=0}^T \bar{L}^4(t) \right) / n(T)$$

Def. 3.22. Sei $c(t)$ die Anzahl der Seiten des Primärspeichers, die im Zeitintervall $[t, t+1)$ von Aufträgen belegt sind, also

$$c(t) := \sum_{i=1}^M \sum_{l=1}^L q_{li}^i(t)$$

Dann ist die m i t t l e r e P r i m ä r - s p e i c h e r b e l e g u n g $\bar{c}(T)$ im Zeitintervall $[0, T]$:

$$\bar{c}(T) := \left(\sum_{t=0}^T c(t) \right) / T$$

Die Definitionen 3.10. und 3.11. legen eine V e r - s u c h s a n o r d n u n g fest, die es gestattet, das Rechenanlagenmodell mit statistischen Methoden zu analysieren. Diese Versuchsanordnung wurde in einem Programm für die Rechenanlage CD 3300 simuliert.

Für die Kenndaten, auf denen die Analyse der Rechenanlage basieren soll, werden Stichproben entnommen, aus denen gemäß den Definitionen 3.12. bis 3.22. Mittelwerte berechnet werden; analog lassen sich Werte für die Streuung berechnen und für jeden dieser Werte ergeben sich in Abhängigkeit von der Anzahl der Stichproben Vertrauensintervalle (benutzt wurden dabei die in Zurmühl (1965) angegebenen Methoden). Die so errechneten Werte sind abhängig von der Dauer T der Simulation eines Experimentes und vom Zustand der Versuchsanordnung im Startzeitpunkt. Die Dauer der Experimente, für die in Abschnitt 3.3. Ergebnisse angegeben sind, ist so gewählt, daß Mittelwerte und Streuung der Kenndaten im Sinne der Statistik konstant waren, so daß die angegebenen Ergebnisse Mittelwerte für den Gleichgewichtszustand des simulierten Warteschlangensystems sind.

Die Programme sind so konzipiert, daß die Verteilungsfunktionen $\mathcal{J}(x)$, $\mathcal{f}(x)$, $\mathcal{M}(x)$, $\mathcal{L}(x)$, $\mathcal{U}_{SE}(x)$ und $\mathcal{U}_{TE}(x)$ ohne

Schwierigkeiten jeweils den Anforderungen an die Rechenanlage entsprechend gewählt werden können; ebenso sind die Kapazität M_0 des Primärspeichers und die Abbildungen ρ , x und σ austauschbar.

3.3. Einige Simulationsergebnisse

Im folgenden werden fünf spezielle Strategien angegeben, für die das Verhalten eines Rechenanlagenmodells simuliert wurde. Einige Simulationsergebnisse werden diskutiert.

In dem simulierten Rechenanlagenmodell wurden eine Magnetrommel als Sekundär- und Magnetplatten als Tertiärspeicher eingesetzt. Die Verteilungsfunktionen für die Dauer der Umspeicherung sind den physikalischen Eigenschaften dieser Speicher entsprechend festgelegt. In $\mathcal{U}_{SE}(x|m)$ sind folgende Parameter einer Trommel berücksichtigt:

1. Die Dauer einer vollständigen Trommelumdrehung; in den angegebenen Beispielen beträgt sie 18 ms.
2. Die Zeit, in der ein Wort zwischen der Trommel und dem Primärspeicher umgespeichert werden kann, wenn kein Suchprozeß nötig ist; sie beträgt hier $3,74 \mu s$.

In $\mathcal{U}_{TE}(x|m)$ sind folgende Parameter eines Plattenspeichers berücksichtigt:

1. Die Dauer einer vollständigen Plattenumdrehung; sie beträgt hier 25 ms.
2. Die Anzahl der Zylinder; hier 200.
3. Die Dauer einer Positionierung der Lese- und Schreibköpfe zwischen benachbarten Zylindern ohne Start- und Stopzeit; hier $0,71$ ms.
4. Start- und Stopzeit für eine Positionierung; hier $35,3$ ms.
5. Die Zeit, in der ein Wort zwischen Platten- und Primärspeicher umgespeichert werden kann, wenn kein Suchprozeß nötig ist; sie beträgt im Beispiel $28,8 \mu s$.

Zur Definition von fünf speziellen Strategien geben wir zunächst einige Abbildungen an.

σ war definiert als Abbildung $\sigma: \mathcal{H}(L) \rightarrow \mathbb{Z}_L$. Bezeichnen wir mit X^0 ein beliebiges Element aus $\mathcal{H}(L)$ mit den Komponenten $(x_{lm}^0)_{L6}$, mit $I(X^0)$ bzw. $I'(X^0)$ die durch

$$I(X^0) := \{l \mid l \in \{1, \dots, L\} \wedge x_{l4}^0 \neq 0\} \text{ bzw. } \\ I'(X^0) := \{l \mid l \in I(X^0) \wedge (\forall k(k \in I(X^0)) \Rightarrow x_{l4}^0 \leq x_{k4}^0)\}$$

für $X^0 \neq 0$ definierten Indexmengen.

Dann sei

$$q_1(X^0) := \begin{cases} 1 & \text{für } X^0 \neq 0 \\ 0 & \text{für } X^0 = 0 \end{cases} \\ q_2(X^0) := \begin{cases} i & \text{falls genau ein } i \in \{1, \dots, L\} \text{ existiert} \\ & \text{mit } x_{i4}^0 \neq x_{i6}^0 \\ 0 & \text{für } X^0 = 0 \\ \min I'(X^0) & \text{sonst} \end{cases}$$

Hierzu sei bemerkt, daß nach Definition der Zustandsfolgen in der Untermenge von $\mathcal{H}(L)$, die $Q^4(t)$ durchlaufen kann, höchstens ein Zeilenindex i mit $q_{i4}^4(t) \neq q_{i6}^4(t)$ existieren kann.

$$q_3(X^0) := \begin{cases} \min I'(X^0) & \text{falls } X^0 \neq 0 \\ 0 & \text{falls } X^0 = 0 \end{cases}$$

x war definiert als Abbildung $x: \mathcal{H}(2L) \rightarrow \mathbb{Z}_{2L}$. Mit X^1 und X^2 bezeichnen wir beliebige Elemente aus $\mathcal{H}(L)$ mit den Komponenten $(x_{lm}^1)_{L6}$ bzw. $(x_{lm}^2)_{L6}$ und $\begin{pmatrix} X^1 \\ X^2 \end{pmatrix}$ mit $X^0 \cdot I(X^1)$ bzw. $I'(X^1)$ seien die durch

$$I(X^1) := \{l \mid l \in \{1, \dots, L\} \wedge x_{l4}^1 \neq 0\} \text{ bzw. } \\ I'(X^1) := \{l \mid l \in I(X^1) \wedge (\forall k(k \in I(X^1)) \Rightarrow x_{l4}^1 \leq x_{k4}^1)\}$$

für $X^1 \neq 0$ definierten Indexmengen.

Dann seien

$$x_1(X^0) := \begin{cases} 1 & \text{falls } x_{15}^2 = 0 \text{ und } x_{15}^1 \neq 0 \\ & \text{oder } x_{15}^1 \neq x_{16}^1 \\ L+1 & \text{falls } x_{15}^2 \neq 0 \text{ und } x_{15}^1 = x_{16}^1 \\ 0 & \text{sonst} \end{cases}$$

$$x_i(X^0) := \begin{cases} L+1 & \text{falls } x_{15}^2 \neq 0 \text{ und } x_{15}^1 = x_{16}^1 \\ & \text{für alle } l \in \{1, \dots, L\} \\ \min I'(X^1) & \text{falls } x_{15}^2 = 0 \text{ und } X^1 \neq 0 \text{ und} \\ & x_{15}^1 = x_{16}^1 \text{ für alle } l \in \{1, \dots, L\} \\ i & \text{falls genau ein } i \in \{1, \dots, L\} \\ & \text{mit } x_{15}^1 \neq x_{16}^1 \text{ existiert} \\ 0 & \text{sonst} \end{cases}$$

ρ war definiert als Abbildung $\rho: \mathcal{H}(4L) \rightarrow \mathcal{H}(4L)$. Da die Primärspeicherverwaltung im allgemeinen mehrere Aufträge, die in der P - Schlange stehen, gleichzeitig abfertigt, erscheint es angemessen, spezielle Abbildungen durch Angabe eines Flußdiagrammes zu definieren.

Für $i \in \{1, \dots, 4\}$ sei X^i ein beliebiges Element aus $\mathcal{H}(L)$ mit den Zeilenvektoren ρ_1^i für $l \in \{1, \dots, L\}$ und den Komponenten $(x_{1m}^i)_{L6}$. $L(X^i)$ sei der durch

$L(X^i) := \min \{ l | l \in \{1, \dots, L\} \wedge \rho_1^i = 0 \}$ definierte Zeilenindex.

Ist $X^0 := \begin{pmatrix} X^1 \\ X^2 \\ X^3 \\ X^4 \end{pmatrix}$, so werden die Abbildungen durch

Flußdiagramme definiert, die zu jedem $X^0 \in \mathcal{H}(4L)$ das Bild $\rho(X^0)$ liefern; die Bilder 11, 12 und 13 geben die Definition von g_1 , g_2 und g_3 an.

Zwei der speziellen Abbildungen - g_1 und g_2 - gestatten keine Verdrängungen. Dies hat zur Folge, daß der Sekundärspeicher für diese Fälle nicht benutzt wird, so daß q_{12}^i für alle $i \in \{1, \dots, 4\}$ und alle $l \in \{1, \dots, L\}$ irrelevant ist.

Ausgehend von diesen Abbildungen, werden fünf ablaufsteuernde Strategien definiert:

1. $S_{111} := (g_1, x_1, \alpha_1)$ - Kurzbezeichnung: FIFO

Primärspeicher - Verwalter und Rechnerkern führen alle Teil - Aufträge in der Reihenfolge ihrer Ankunft in der P - bzw. R - Schlange aus.

Das E/A - Steuerwerk bevorzugt die Teil - Aufträge der PH - Schlange, ohne Ausführungsunterbrechungen zu erlauben; die Teil - Aufträge derselben Warteschlange (PH - und HP - Schlange) werden in der Reihenfolge ihrer Ankunft abgearbeitet.

2. $S_{211} := (g_2, x_1, \alpha_1)$ - Kurzbezeichnung MSBP

Der Primärspeicher - Verwalter bevorzugt die Teil - Aufträge mit minimalem Speicherbedarf und beschränkt sich bei der Anordnung der Aufträge auf die P - Schlange.

Das E/A - Steuerwerk bevorzugt die Teil - Aufträge der PH - Schlange, ohne Ausführungsunterbrechungen zu erlauben; die Teil - Aufträge derselben Warteschlange (PH- und HP - Schlange) werden in der Reihenfolge ihrer Ankunft abgearbeitet.

Der Rechnerkern führt die Teil - Aufträge in der Reihenfolge ihrer Ankunft in der R - Schlange aus.

3. $S_{112} := (g_1, x_1, \alpha_2)$ - Kurzbezeichnung: KADR

Der Primärspeicher - Verwalter führt die Teil - Aufträge in der Reihenfolge ihrer Ankunft in der P - Schlange aus.

Das E/A - Steuerwerk bevorzugt die Teil - Aufträge der PH - Schlange, ohne Ausführungsunterbrechungen zu erlau-

ben; die Teil - Aufträge derselben Warteschlange (PH - und HP - Schlange) werden in der Reihenfolge ihrer Ankunft abgearbeitet.

Der Rechnerkern bevorzugt die Teil - Aufträge mit der kürzesten virtuellen Ausführungsdauer und gestattet dabei keine Ausführungsunterbrechungen.

4. $S_{212} := (q_2, x_1, \sigma_2)$ - Kurzbezeichnung MSPKAR

Der Primärspeicher - Verwalter bevorzugt die Teil - Aufträge mit minimalem Speicherbedarf und beschränkt sich bei der Anordnung der Aufträge auf die P - Schlange. Das E/A - Steuerwerk bevorzugt die Teil - Aufträge der PH - Schlange, ohne Ausführungsunterbrechungen zu erlauben; die Teil - Aufträge derselben Warteschlange (PH - und HP - Schlange) werden in der Reihenfolge ihrer Ankunft abgearbeitet.

Der Rechnerkern bevorzugt die Teil - Aufträge mit der kürzesten virtuellen Ausführungsdauer und gestattet dabei keine Ausführungsunterbrechungen.

5. $S_{323} := (q_3, x_2, \sigma_3)$ - Kurzbezeichnung KADPKR

Primärspeicher - Verwalter, E/A - Steuerwerk und Rechnerkern bevorzugen die Teil - Aufträge mit kürzester virtueller Ausführungsdauer. Die Anordnung der Aufträge durch den Primärspeicher - Verwalter erstreckt sich auf die Teil - Aufträge in der P -, der HP - und der R - Schlange. In Abhängigkeit von der virtuellen Ausführungsdauer werden Teil - Aufträge aus der HP - in die P - Schlange verdrängt, indem ihnen der von ihnen belegte Primärspeicherraum entzogen wird. Teil - Aufträgen der R - Schlange kann der Primärspeicherraum ebenfalls entzogen werden; das entsprechende Programm und die zugehörigen Daten werden in den Sekundärspeicher übertragen (über die PH - Schlange); nach der Umspeicherung gelangen diese Teil - Aufträge in die P - Schlange. Der Primärspeicher - Verwalter führt Ausführungsunterbrechungen

bei Umspeicherprozessen zwischen Sekundär- und Primärspeicher, sowie bei Operationen des Rechnerkerns durch. Das E/A - Steuerwerk bevorzugt die Teil - Aufträge der PH - Schlange; sie werden in der Reihenfolge ihrer Ankunft in der PH - Schlange ausgeführt. Die Teil - Aufträge in der HP - Schlange werden geordnet nach ihrer virtuellen Ausführungsdauer abgearbeitet.

Der Rechnerkern bevorzugt die Teil - Aufträge mit kürzester virtueller Ausführungsdauer und läßt dabei Unterbrechungen zu.

In Bild 14 ist für die fünf Strategien die mittlere Verweilzeit eines Auftrages bei fester Primärspeicherkapazität $M_0 = 160 \cdot 1024$ Wörter als Funktion der Anzahl NA der aktiven Benutzer dargestellt. Mit dem Parameter NA lassen sich insbesondere die mittlere Auslastung des Rechnerkerns und des E/A - Steuerwerkes variieren. Für NA = 40 war es für die Strategien FIFO und MSBP nicht mehr mit vertretbarem Rechenzeit- und Speicheraufwand möglich, die mittlere Verweilzeit zu berechnen.

Von besonderem Interesse ist die Analyse der mittleren Wartezeit eines Auftrages bei fester Anzahl NA der aktiven Benutzer und variabler Primärspeicherkapazität. In den Bildern 15, 16, 17, 18 und 19 sind diese, aufgeschlüsselt nach Wartezeit auf Primärspeicherzuteilung, HP - Umspeicherung, Ausführung durch den Rechnerkern und PH - Umspeicherung für alle Strategien dargestellt. Jeweils markiert ist die Speicherkapazität, für die erstmals die Wartezeit auf Primärspeicherzuteilung den Wert 0 annimmt.

Für die Strategien FIFO, MSBP, KADR und MSPKAR ist die reale Ausführungsdauer eines Auftrages, unabhängig von der Strategie, festgelegt durch die virtuelle Ausfüh-

rungsdauer, die Dauer der HP - Umspeicherung und die Dauer der PH - Umspeicherung. Ihr Mittelwert ist $\bar{r} = 1,3$ sec und ergibt sich aus den Summanden:

Mittelwert der virtuellen Ausführungsdauer:	785 ms
Mittlere Dauer einer HP - Umspeicherung:	300 ms
Mittlere Dauer einer PH - Umspeicherung:	215 ms

Für die Strategie KADPKR ist die reale Ausführungsdauer abhängig von der Primärspeicherkapazität; die Wahrscheinlichkeit der Verdrängung eines Auftrages aus dem Primärspeicher wächst mit jeder Reduktion der Primärspeicherkapazität. In Bild 20 ist die mittlere Anzahl der vollständig ausgeführten HP - Umspeicherprozesse für $N_A = 40$ als Funktion von M_0 dargestellt. Der Einfluß dieser Umspeicherprozesse auf die reale Ausführungsdauer ist für den als Beispiel gewählten Sekundärspeicher noch nicht wesentlich.

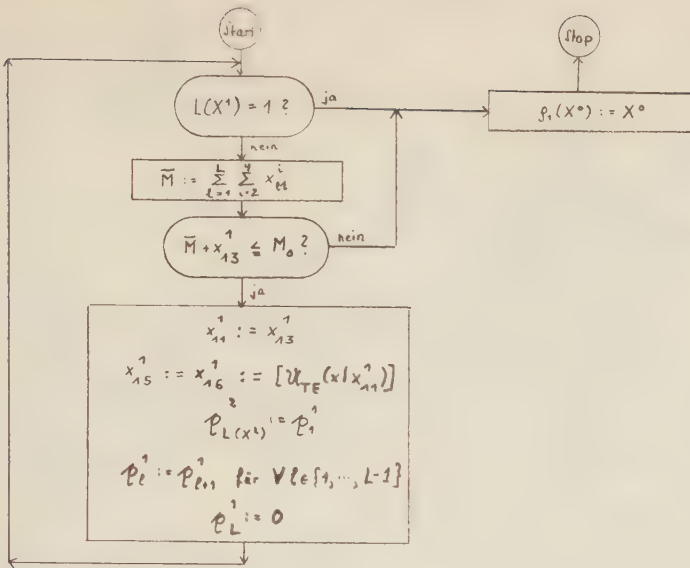


Bild 11: Die Strategie g_1

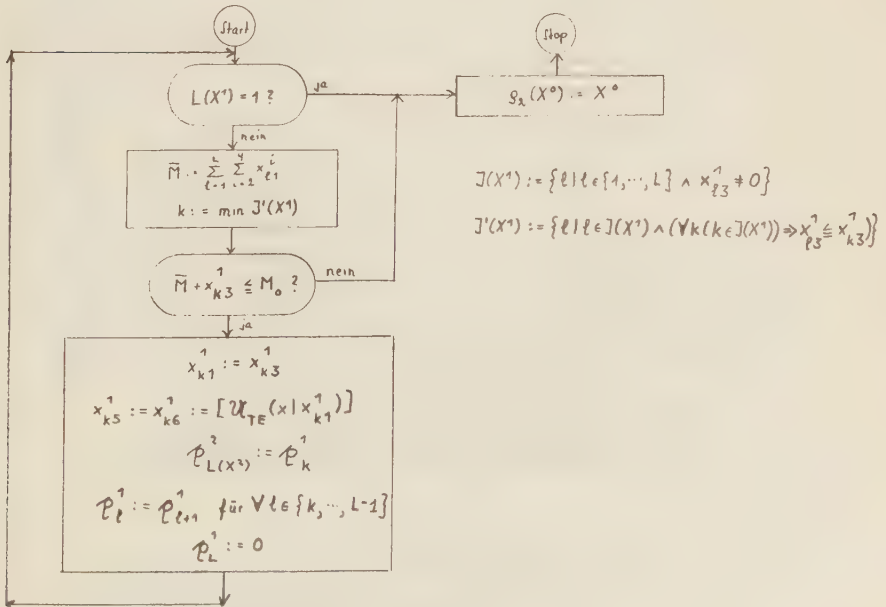
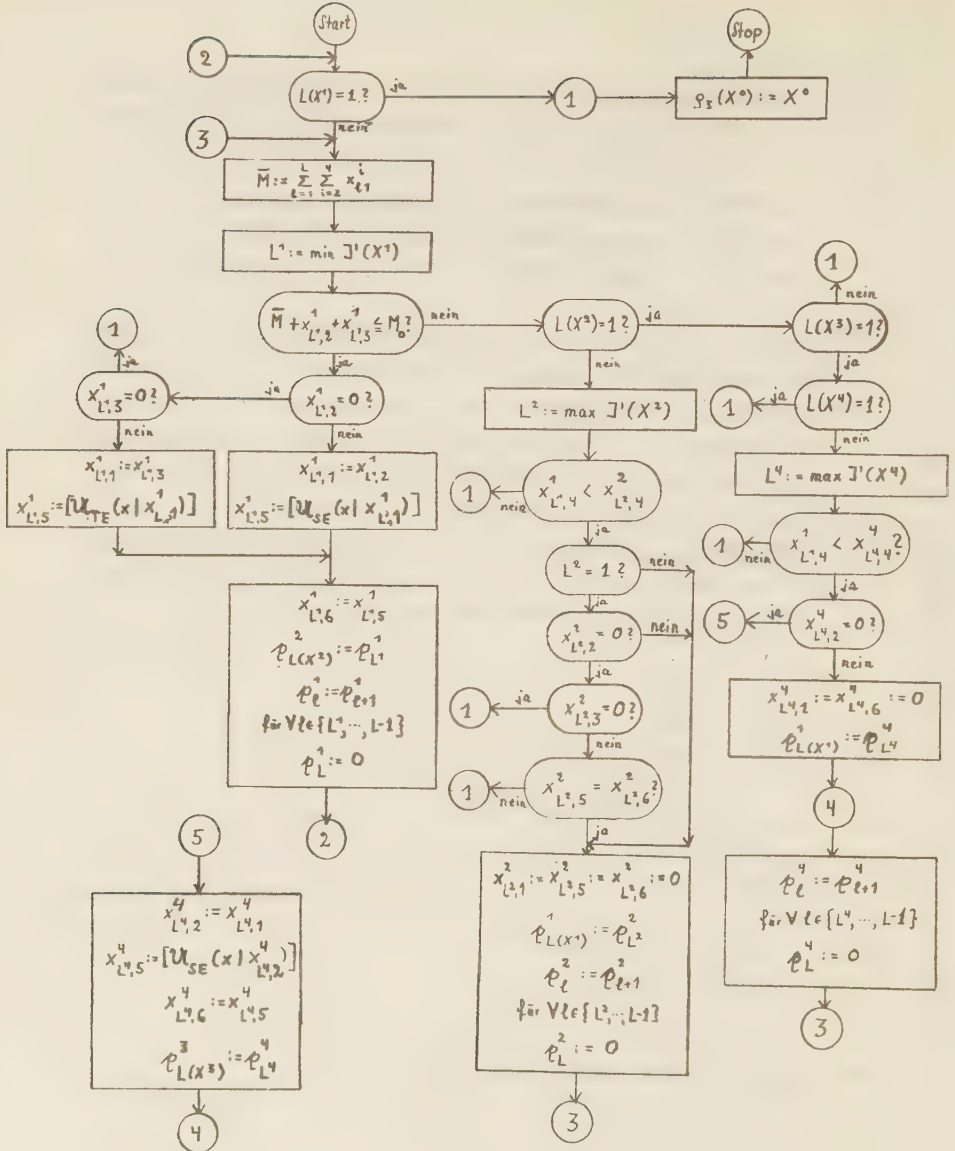


Bild 12: Die Strategie g_2



Für $i \in \{1, 2, 3, 4\}$: $J(X^i) := \{l | l \in \{1, \dots, L\} \wedge x_{li}^i \neq 0\}$

$J^1(X^1) := \{l | l \in J(X^1) \wedge (\forall k (k \in J(X^1)) \Rightarrow x_{l4}^1 \leq x_{k4}^1)\}$

Für $i \in \{2, 3, 4\}$: $J^i(X^i) := \{l | l \in J(X^i) \wedge (\forall k (k \in J(X^i)) \Rightarrow x_{li}^i \geq x_{ki}^i)\}$

Bild 13: Die Strategie 93

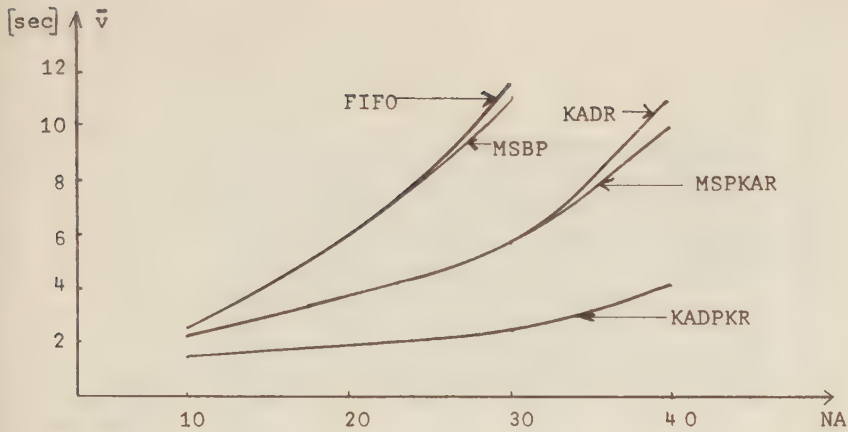
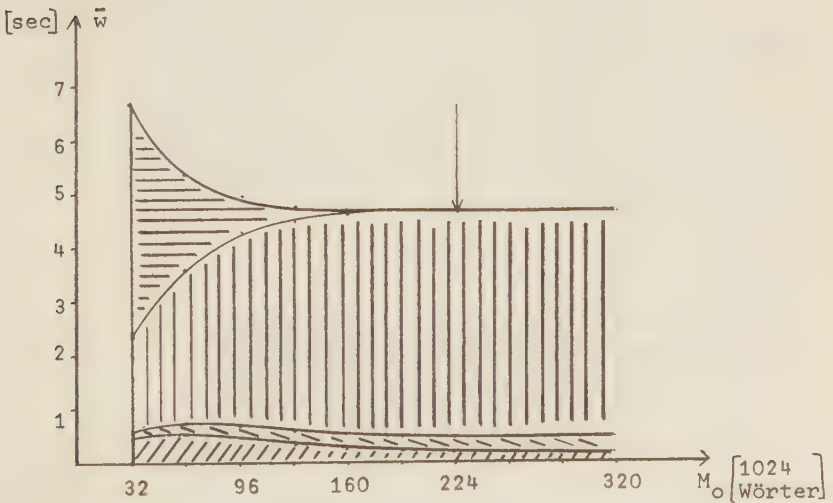


Bild 14: Mittlere Verweilzeit als Funktion von NA und S bei konstanter Primärspeicherkapazität $M_0 = 160 \cdot 1024$ Wörter.



Wartezeit auf: $\equiv$ P - Zuteilung, $|||$ RK - Ausführung,
 $///$ HP - Umspeicherung, $\\$ PH - Umspeicherung.

Bild 15: Analyse der mittleren Wartezeit als Funktion der Primärspeicherkapazität M_0 für NA = 20 und die Strategie FIFO

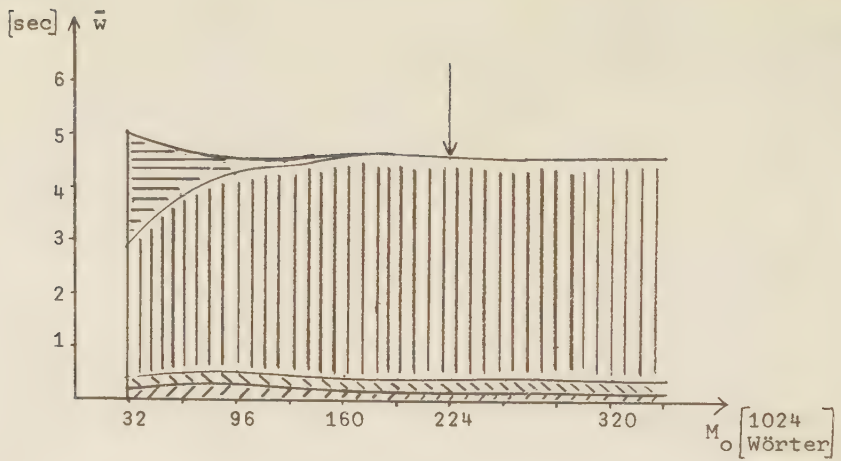


Bild 16: Analyse der mittleren Wartezeit als Funktion der Primärspeicherkapazität M_0 für $NA=20$ und die Strategie MSBP.

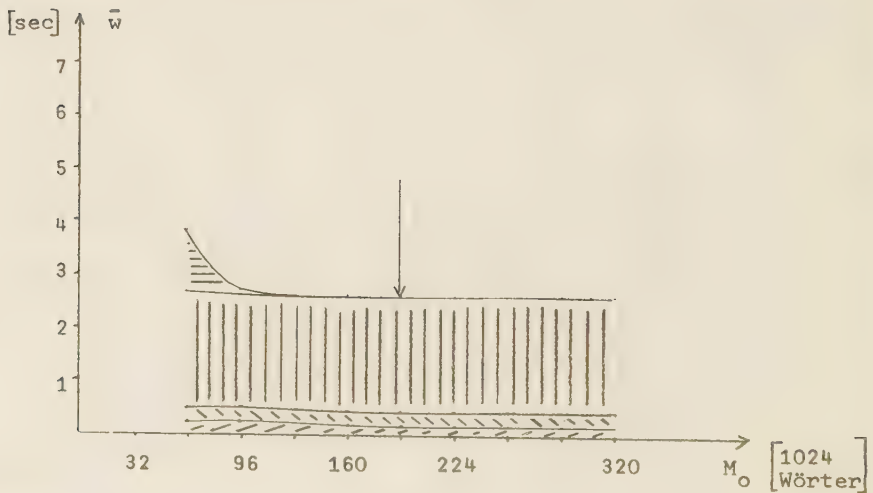


Bild 17: Analyse der mittleren Wartezeit als Funktion der Primärspeicherkapazität M_0 für $NA=20$ und die Strategie KADR.

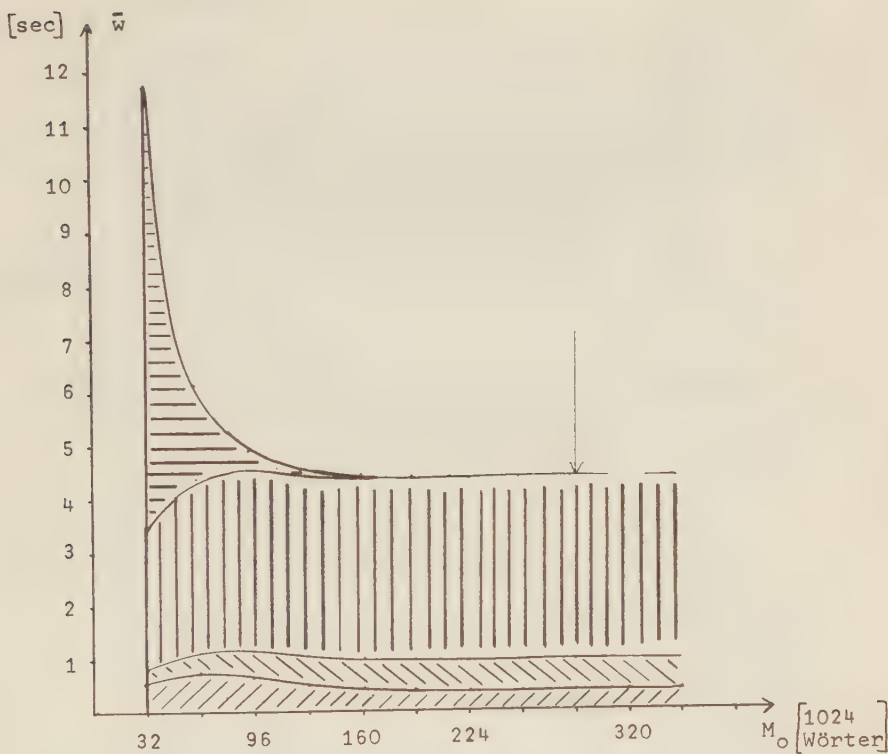


Bild 18: Analyse der mittleren Wartezeit als Funktion der Primärspeicherkapazität M_o für $NA=30$ und die Strategie MSPKAR.

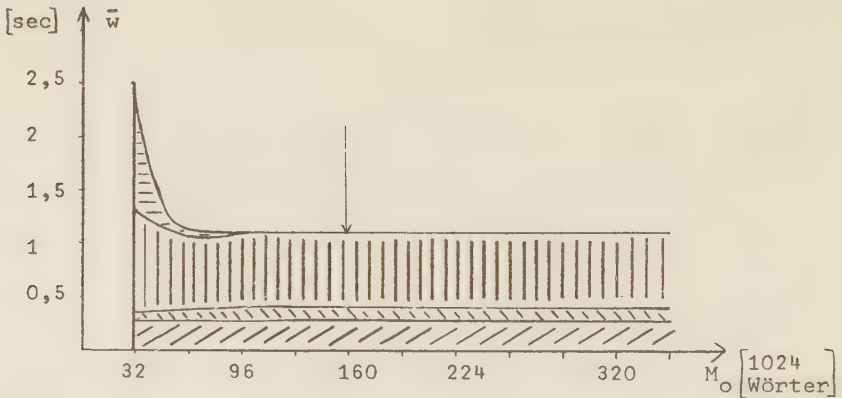


Bild 19: Analyse der mittleren Wartezeit als Funktion der Primärspeicherkapazität M_o für $NA=30$ und die Strategie KADPKR.

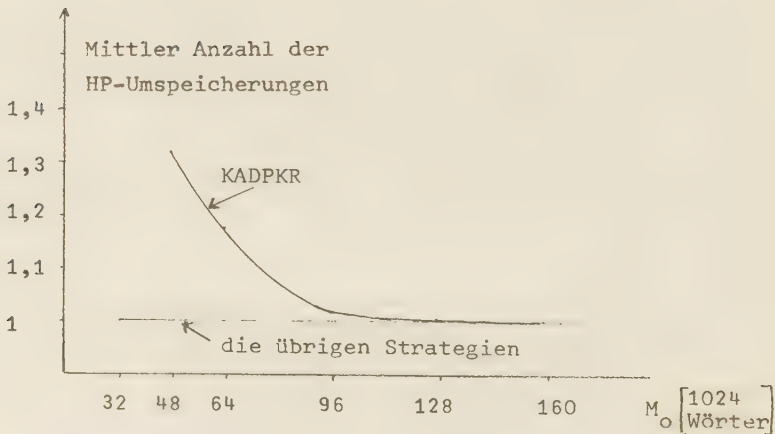


Bild 20: Mittlere Anzahl der HP-Umspeicherungen als Funktion der Primärspeicherkapazität M_o für $NA=40$ und die Strategie KADPKR.

4. Speicherstrategien

4.0. Vorbemerkungen

Wie in Kapitel 3 ausführlich dargelegt ist, hat die Primärspeicher - Verwaltung und insbesondere die Anzahl und die Dauer von Umspeicherprozessen einen wesentlichen Einfluß auf die reale Ausführungsdauer eines Auftrages. Dieses Erkenntnis führte zur Konzeption der **S e i t e n - e i n t e i l u n g** im Primärspeicher.

Diese Methode (für eine ausführliche Diskussion sei auf Spies (1967) verwiesen) hat bzgl. der Speicherverwaltung im wesentlichen zwei Eigenschaften:

1. Einem Auftrag können - unabhängig von den Adressen in den Befehlen des entsprechenden Programmes - Seiten im Primärspeicher zugeteilt werden, deren Seitennummer während des Programmlaufes dynamisch festgelegt wird. Diese Eigenschaft ist in Kapitel 3 ausgenutzt.
2. Einem Auftrag muß nicht notwendig der gesamte von ihm angeforderte Speicherraum vor Beginn der Ausführung zugeteilt werden. Die einzelnen Seiten können dynamisch zugeteilt werden in Abhängigkeit von den Operationen, die gerade im Rechnerkern oder im E/A - Steuerwerk auszuführen sind.

In diesem Kapitel werden Untersuchungen durchgeführt über die Möglichkeiten, die zweite dieser Eigenschaften zu nutzen. Wir beschränken uns dabei auf die Speicherverwaltung bzgl. eines Auftrages und auf den Auftrags - Raum eines Programmes.

In Abschnitt 4.1. wird ausgeführt, wie eine Beschreibung der Programmläufe aus dem Auftrags - Raum des entsprechenden Programmes abgeleitet werden kann, wenn eine Zerlegung des Speichers der virtuellen Anlage vorgegeben ist. Die resultierende Auftrags - Raum - Beschreibung kann Unter-

suchungen zur Strukturierung des Speicherraumes einer realen Rechenanlage und zur Analyse von Strategien der Speicherverwaltung zugrunde gelegt werden.

In Abschnitt 4.2. beschränken wir uns auf Speicher mit einer Seitenstruktur mit Seiten konstanter Größe. Wenn sich die Programmläufe eines Auftrags - Raumes beschreiben lassen als ungetaktete Markoff - Kette, ergeben sich besonders übersichtliche Verhältnisse. Für diesen Fall werden drei Speicherstrategien miteinander verglichen in Bezug auf den bei Programmläufen im mittel belegten Speicherraum und die mittlere Anzahl der Aufrufe von Befehlen und Daten, die nicht im Primärspeicher stehen.

4.1. Der Auftrags - Raum

Sei $Z = \mathbb{Z}_1 \times \mathbb{Z}_1 \times \mathbb{Z}_n \times \mathbb{Z}_G$ die Zustandsmenge der virtuellen Rechenanlage und $(\Omega, \mathcal{P}(\Omega), P)$ ein Auftrags - Raum bzgl. eines Programmes π . In keinem der Programmläufe trete eine Ein- oder Ausgabeoperation auf. Diese Annahme ist keine prinzipielle Einschränkung; sie dient der Übersichtlichkeit der nachfolgenden Ausführungen und kann durch Erweiterung der in Def. 4.4. angegebenen Zustandsmenge aufgehoben werden.

Sei $Y := \{Y_1, \dots, Y_M\}$ eine Zerlegung der Menge $\{1, 2, \dots, N\}$; es gelte also $\bigcup_{m=1}^M Y_m = \{1, \dots, N\}$ und $Y_n \cap Y_m = \emptyset$ für alle $n, m \in \{1, \dots, M\}$ mit $n \neq m$.

Def. 4.1. Für $Z' := \mathbb{Z}_1 \times \mathbb{Z}_1 \times \mathbb{Z}_n \times \mathbb{Z}_G$ sei $\varphi : Z \rightarrow Z'$ die durch

$$\varphi(z_1, z_2, 0, z_4) := (z_1, z_2, 0, z_4) \text{ für alle } (z_1, z_2, z_4) \in \mathbb{Z}_1 \times \mathbb{Z}_1 \times \mathbb{Z}_G \text{ und}$$

$$\varphi(z_1, z_2, z_3, z_4) := (z_1, z_2, m, z_4) \text{ für alle } z_3 \in Y_m \text{ und alle } (z_1, z_2, z_4) \in \mathbb{Z}_1 \times \mathbb{Z}_1 \times \mathbb{Z}_G \text{ definierte Abbildung. } Z' \text{ ist die Zustandsmenge}$$

der virtuellen Rechen-
anlage bzgl. der Zerle-
gung Y

Entsprechend der durch ϕ festgelegten Strukturierung von Z ergeben sich Veränderungen in der Beschreibung der Programmläufe.

Def. 4.2. Sei $\tilde{\Omega}$ die Menge der Abbildungen $\tilde{\omega} : \mathbb{R}^+ \rightarrow Z'$.
 $\tilde{\psi} : \Omega \rightarrow \tilde{\Omega}$ sei die durch
 $\tilde{\psi}(\omega) := \tilde{\omega} \Leftrightarrow (\forall t(t \in \mathbb{R}^+) \Rightarrow \tilde{\omega}(t) = \phi(\omega(t)))$ für
alle $\omega \in \Omega$ definierte Abbildung.

Bezeichnen wir mit Ω' die Menge der Bildelemente von Ω
bzgl. $\tilde{\psi}$ und mit ψ' die Abbildung $\psi' : \Omega \rightarrow \Omega'$, die für alle
 $\omega \in \Omega$ mit $\tilde{\psi}$ übereinstimmt, so ist ψ' eine meßbare Abbildung
bzgl. $(\Omega, \mathcal{P}(\Omega))$ und $(\Omega', \mathcal{P}(\Omega'))$, so daß ψ das Wahrschein-
lichkeitsmaß P in ein Wahrscheinlichkeitsmaß P' abbildet.

Def. 4.3. Ist Ω' die Menge der Bildelemente von Ω bzgl. $\tilde{\psi}$
und P' das Bild von P bzgl. ψ' , so ist
 $(\Omega', \mathcal{P}(\Omega'), P')$ der Auftrags - Raum
von $\mathcal{W}$ bzgl. der Zerlegung Y .

Für die Untersuchung der Programmläufe ist es zweckmäßig,
eine weitere Vereinfachung vorzunehmen.

Def. 4.4. Sei $X := \{X^0, X^1, \dots, X^M\}$ und $\omega' \in \Omega'$. Der Programm-
lauf ω' befinde sich zum Zeitpunkt $t \in \mathbb{R}^+$ im
Zustand $X^0 : \Leftrightarrow \omega'(t) = (0, 0, 0, 0)$. Der
Programmlauf ω' befinde sich zum Zeitpunkt t
im Zustand X^m mit $m \in \{1, \dots, M\}$
 $: \Leftrightarrow \omega'(t) = (1, 0, m, 0) \vee$
 $\omega'(t) = (1, 0, 0, 0) \wedge$
 $(\exists t'(t' \in \mathbb{R}^+ \wedge t' < t \wedge \omega'(t') = (1, 0, m, 0) \wedge$
 $(\forall t''(t'' \in (t', t] \Rightarrow \omega'(t'') = (1, 0, 0, 0))))$

Gemäß dieser Zustandsdefinition kann jeder Programmauf-
 lauf eindeutig beschrieben werden durch eine Folge von
 Zuständen $X^{(1)}, X^{(2)}, X^{(3)} \dots$ mit $X^{(i)} \in \{X^0, \dots, X^M\}$ und
 $X^{(i)} \neq X^{(i+1)}$ für alle $i \in \mathbb{N}$ und einer Folge von halboffen-
 en Intervallen $[0, t_1), [t_1, t_2), [t_2, t_3) \dots$. Befindet sich
 ein Programmauf- ω zum Zeitpunkt t im Zustand $X^{(i)} = X^m$
 mit $m \neq 0$, so wird während des ganzen Zeitintervalles
 $[t_{i-1}, t_i)$ kein Speicherwort aus $\{1, \dots, N\} - \{Y_m\}$ angespro-
 chen; andererseits ist der Y_m entsprechende Speicherraum
 während dieses Zeitintervalles nicht entbehrlich, da
 wenigstens eine der in Y_m enthaltenen Zellen angespro-
 chen wird.

Geht man von einem Auftrags - Raum $(\Omega, \mathcal{P}(\Omega), P)$ aus, so
 kann man für verschiedene Zerlegungen von $\{1, \dots, N\}$ durch
 Analyse der entsprechenden Auftrags - Räume die Zweck-
 mäßigkeit der Zerlegungen beurteilen. Andererseits kann
 man - ausgehend von einem Auftrags - Raum bzgl. einer
 festen Zerlegung - Strategien der Verwaltung eines Spei-
 chers, der entsprechend der Zerlegung strukturiert ist,
 untersuchen. Für die Methode der Seiteneinteilung führen
 wir im nächsten Abschnitt derartige Untersuchungen durch.

4.2. Drei spezielle Speicherstrategien

Wir nehmen an, daß im Speicher der virtuellen Rechan-
 lage und im Primärspeicher der realen Rechananlage eine
 S e i t e n e i n t e i l u n g durchgeführt sei. Jede
 Seite enthalte L Wörter.

Def. 4.5. Für $n \in \mathbb{Z}_N$ sei $\left[\frac{n}{L}\right]$ die kleinste ganze Zahl, für
 die $\left[\frac{n}{L}\right] \geq \frac{n}{L}$ gilt und für $Z' := \mathbb{Z}_1 * \mathbb{Z}_1 * \mathbb{Z}_{\left[\frac{N}{L}\right]} * \mathbb{Z}_G$
 sei $\varphi : Z \rightarrow Z'$ die durch
 $\varphi(z_1, z_2, z_3, z_4) := (z_1, z_2, \left[\frac{z_3}{L}\right], z_4)$ für alle
 $(z_1, z_2, z_3, z_4) \in Z$ definierte Abbildung.

Der Auftrags - Raum bzgl. der durch die Seiteneinteilung definierten Zerlegung, der sich nach Def. 4.3. unter Verwendung der in Def. 4.5. angegebenen Abbildung φ ergibt, sei wieder mit $(\Omega, \mathcal{P}(\Omega), P)$ bezeichnet und allen folgenden Ausführungen zugrunde gelegt.

Def. 4.6. Für $\omega \in \Omega$ seien

$$M(\omega) := \max \{ z_3 | \exists t (t \in \mathbb{R}^+ \wedge \omega(t) = (1, 0, z_3, 0)) \}$$

$$\text{und } M := \max \{ M(\omega) | \omega \in \Omega \}$$

M ist die maximale Seitenanzahl, die ein Programmauf aus Ω anfordern kann.

Für die Untersuchungen der Programmläufe bzgl. π können wir uns folglich auf die Zustandsmenge $X := \{x^0, \dots, x^M\}$ beschränken. Wir nehmen an, daß Aufträge bzgl. π auszuführen sind und vor Beginn jedes Programmaufes das Programm und die entsprechenden aktuellen Parameter im Hintergrundspeicher stehen. Für die Primärspeicherzuteilung werde eine der drei Strategien g_1' , g_2' oder g_3' benutzt.

g_1' : Die Strategie g_1' lege fest, daß jedem Programmauf vor dessen Beginn der gesamte Speicherbedarf des entsprechenden Programmes zugeteilt wird.

g_2' : Die Strategie g_2' lege fest, daß jedem Programmauf während seiner gesamten Ausführung genau eine Seite im Primärspeicher zugeteilt wird und alle Befehle und Daten bei Bedarf in dieser Seite gespeichert werden.

g_3' : Die Strategie g_3' lege fest, daß jedem Programmauf so viele Seiten im Primärspeicher zugeteilt werden, wie er während seiner Ausführung anfordert, jedoch jede Seite erst zum aktuellen Zeitpunkt ihrer Anforderung.

Es sei bemerkt, daß diese Strategien mit den in Kapitel 3 angegebenen insofern unvergleichbar sind, als dort Speicher-verwaltung im Rahmen der Organisation der gesamten Rechen-

anlage betrachtet wurde, während hier die Speicherzu-
teilung bzgl. eines einzelnen Programmlaufes untersucht
wird.

Für die B e w e r t u n g einer Strategie werden zwei
charakteristische Parameter untersucht: die maximale
Anzahl W der Seiten, die während eines Programmlaufes
im Primärspeicher gleichzeitig belegt sind, und die An-
zahl K der Fälle, in denen ein Umspeicherprozeß zwischen
Hintergrund- und Primärspeicher vor und während eines
vollständigen Programmlaufes notwendig ist. W und K sind
in Abhängigkeit von der Speicherstrategie und dem Auf-
trags - Raum i. a. stochastische Variable, deren Mittel-
werte für eine spezielle Programm - Raum - Klasse angege-
ben werden.

Aus der Definition von g_1^1 und den Voraussetzungen über den
Auftrags - Raum ergeben sich unmittelbar $W = M$ und $K = 1$;
es ist ein Umspeicherprozeß vor Beginn jedes Programmlau-
fes nötig.

Aus der Definition von g_2^1 ergibt sich unmittelbar $W = 1$.

Aus der Definition von g_3^1 ergibt sich $W = K$.

Für die weitere Analyse betrachten wir zunächst den
Fall $M = 2$.

Jeder Programmlauf aus Ω ist zu beschreiben durch eine
Zustandsfolge $X^{(1)}, X^{(2)}, X^{(3)}, \dots$ mit $X^{(i)} \in \{X^0, X^1, X^2\}$
und eine Intervallfolge $[0, t_1), [t_1, t_2), [t_2, t_3), \dots$

Aus der Def. 1.3. für virtuelle Programmläufe folgt, daß
jeder Lauf im Zustand X^0 endet. Wir wollen zusätzlich an-
nehmen, daß jeder Programmlauf im Zustand X^1 beginnt.

Dann ergibt sich folgendes Verhalten: Ein beliebiger Pro-

grammlauf beginnt im Zustand X^1 ; nach einer Ausführungszeit t_1 endet der Programmlauf (d. h. er geht in den Zustand X^0 über) oder er geht in den Zustand X^2 über; nach einer Ausführungszeit $t_2 - t_1$ endet der Programmlauf oder er geht wieder in den Zustand X^1 über usw. Bild 21 zeigt dieses Verhalten in einem Graphen mit 3 Zuständen.

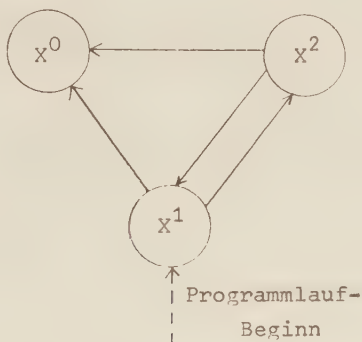


Bild 21: Zustandsgraph der Programmläufe für $M = 2$

Es gelte:

- (1) Die Länge E_1 bzw. E_2 der Zeitintervalle, während der ein beliebiger Programmlauf im Zustand X^1 bzw. X^2 verweilt, sei ausschließlich vom Zustand abhängig und genüge für alle Ausführungsphasen der bedingten Verteilungsfunktion $f(x|1)$ bzw. $f(x|2)$.
- (2) Nach Abschluß jeder Programmlaufphase im Zustand X^1 bzw. X^2 ende der Programmlauf mit der Wahrscheinlichkeit p_1 bzw. p_2 ; mit der Wahrscheinlichkeit $1-p_1$ bzw. $1-p_2$ schließe sich eine Phase im Zustand X^2 bzw. X^1 an. Diese Übergangswahrscheinlichkeiten seien ausschließlich von dem Zustand abhängig, bzgl. dessen eine Programmlaufphase beendet wird.
- (3) Jeder Programmlauf beginne im Zustand X^1 .

Es sei darauf hingewiesen, daß diese Beschreibung der Programmläufe formal übereinstimmt mit den Festlegungen, die in Abschnitt 2.2.2. getroffen wurden. Ist insbesondere $f(x)$ die Verteilungsfunktion der virtuellen Ausführungsdauer von π , so stellt Satz 2.11. den Zusammenhang zwischen $f(x)$, $f(x|1)$ und $f(x|2)$ her.

Unter den Voraussetzungen (1), (2) und (3) untersuchen wir g'_2 und g'_3 .

Strategie g'_2 :

Für den Fall, daß p_1 und p_2 nicht gleichzeitig den Wert 0 annehmen, kann K Werte aus $\mathbb{N}$ annehmen, und zwar gelten: für $n \in \mathbb{N}$ und $n = 2m + 1$:

$$\text{Wahrscheinlichkeit } (K = n) = (1-p_1)^m (1-p_2)^m p_1;$$

für $n \in \mathbb{N}$ und $n = 2m$:

$$\text{Wahrscheinlichkeit } (K = n) = (1-p_1)^m (1-p_2)^{m-1} p_2.$$

Daraus ergibt sich für $p_1 p_2 < p_1 + p_2$ für K der Mittelwert:

$$\bar{K} = \frac{2-p_1}{p_1+p_2-p_1 p_2}$$

Strategie g'_3 :

Nach der Definition von g'_3 kann W Werte aus $\{1, 2\}$ annehmen, wobei gelten:

$$\text{Wahrscheinlichkeit } (W=1) = \text{Wahrscheinlichkeit } (K=1) = p_1$$

$$\text{Wahrscheinlichkeit } (W=2) = \text{Wahrscheinlichkeit } (K=2) = 1-p_1$$

Daraus ergeben sich die Mittelwerte: $\bar{W} = \bar{K} = 2-p_1$

Wir verallgemeinern die Untersuchungen auf $M > 2$. Dazu gelte:

- (1') Die Länge E_i der Zeitintervalle, während der ein beliebiger Programmlauf im Zustand X^i mit $i \in \{1, \dots, M\}$ verweilt, sei ausschließlich vom Zustand X^i abhängig und genüge für alle Ausführungsphasen der bedingten Verteilungsfunktion $f(x|i)$.

(2') P sei eine stochastische Matrix mit $M+1$ Zeilen und Spalten

$$P := \begin{pmatrix} p_{00} & p_{01} & \cdot & \cdot & \cdot & p_{0M} \\ p_{10} & p_{11} & \cdot & \cdot & \cdot & p_{1M} \\ \vdots & \vdots & & & & \vdots \\ p_{M0} & p_{M1} & \cdot & \cdot & \cdot & p_{MM} \end{pmatrix}$$

Nach Abschluß jeder Programmlaufphase im Zustand X^i mit $i \in \{1, \dots, M\}$ ende der Programmlauf mit der Wahrscheinlichkeit p_{i0} bzw. es schließe sich eine Programmlaufphase im Zustand X^k mit $k \in \{1, \dots, M\} - \{i\}$ mit der Wahrscheinlichkeit p_{ik} an. Die Übergangswahrscheinlichkeiten seien ausschließlich von dem Zustand abhängig, bzgl. dessen eine Programmlaufphase beendet wird. Für alle $i \in \{1, \dots, M\}$ gelte $p_{ii} = 0$.

(3') Jeder Programmlauf beginne im Zustand X^1 .

Def. 4.7. Ein Auftrags - Raum, dessen Programmläufe durch die Eigenschaften (1'), (2') und (3') festgelegt sind, bildet eine u n g e t a k t e t e M a r k o f f - K e t t e.

Das Verhalten einer ungetakteten Markoff - Kette wird also wechselseitig durch die Matrix P bzw. durch die Verteilungsfunktionen $f(x|i)$ festgelegt.

Für die Untersuchung von Speicherstrategien nach den angegebenen Kriterien ist die Zeit zwischen aufeinanderfolgenden Zustandsübergängen irrelevant, so daß wir uns auf die Untersuchung der durch P definierten (getakteten) Markoff - Kette beschränken können. Führen wir die übliche Äquivalenzklasseneinteilung für die Menge X durch (wir benutzen hier die in Kemeny/Snell (1960) angegebenen Definitionen), so ist $\{X^0\}$ eine ergodische Menge, also X^0 ein absorbierender Zustand.

Wir nehmen für das Folgende an, daß X bezüglich der durch P definierten Relation in genau zwei Äquivalenzklassen $\{X^0\}$ und $\{X^1, \dots, X^M\}$ zerfällt und $\{X^1, \dots, X^M\}$ eine transiente Menge ist. Diese Voraussetzung ist immer dann erfüllt, wenn das Programm π keine Endlos - Schleife enthält. Sind diese Voraussetzungen erfüllt, so lassen sich die Speicherstrategien g_2^1 und g_3^1 nach der Theorie der absorbierenden Markoff - Ketten analysieren.

Def. 4.8. Seien I die Einheitsmatrix mit M Zeilen und Spalten und

$$\tilde{P} := \begin{pmatrix} p_{11} & p_{12} & \cdot & \cdot & p_{1M} \\ p_{21} & p_{22} & \cdot & \cdot & p_{2M} \\ \vdots & \vdots & & & \vdots \\ p_{M1} & p_{M2} & \cdot & \cdot & p_{MM} \end{pmatrix}$$

Dann sei $C := (I - \tilde{P})^{-1}$. C heißt Fundamentalmatrix der absorbierenden Markoff - Kette und habe die Komponenten c_{ij} für $i, j \in \{1, \dots, M\}$.

Die Existenz von C folgt unmittelbar aus der Voraussetzung, daß $\{X^1, \dots, X^M\}$ transiente Menge einer absorbierenden Markoff - Kette ist.

Satz 4.1. Der Auftrags - Raum $(\Omega, \mathcal{P}(\Omega), P)$ bilde eine ungetaktete Markoff - Kette mit der Übergangsmatrix P . Die durch P für $\{X^0, \dots, X^M\}$ definierte Relation teile $\{X^0, \dots, X^M\}$ in die Äquivalenzklassen $\{X^0\}$ und $\{X^1, \dots, X^M\}$, wobei $\{X^1, \dots, X^M\}$ transiente Menge sei. C sei die Fundamentalmatrix der Kette. Dann gilt: $\bar{K}(g_2^1) = \sum_{i=1}^M c_{1i}$.

Dieser Satz ist eine Folgerung aus einem Satz der Theorie der absorbierenden Markoff - Ketten, zu dessen Beweis auf Theorem 3.3.5. in Kemeny/Snell (1960) verwiesen sei.

Satz 4.2. Der Auftrags - Raum $(\Omega, \mathcal{P}(\Omega), P)$ erfülle die Voraussetzungen des Satzes 4.1. Dann gilt:

$$\bar{W}(g'_3) = \bar{K}(g'_3) = 1 + \sum_{i=2}^M \frac{c_{1i}}{c_{ii}}$$

Auch dieser Satz folgt aus einem Satz der Theorie der absorbierenden Markoff - Ketten, zu dessen Beweis auf Theorem 3.5.9. in Kemeny/Snell (1960) verwiesen sei.

Zusammenfassung: Erfüllt der Auftrags - Raum $(\Omega, \mathcal{P}(\Omega), P)$ die Voraussetzung des Satzes 4.1., so gelten:

$$\bar{W}(g'_1) = M; \quad \bar{K}(g'_1) = 1;$$

$$\bar{W}(g'_2) = 1; \quad \bar{K}(g'_2) = \sum_{i=1}^M c_{1i};$$

$$\bar{W}(g'_3) = \bar{K}(g'_3) = 1 + \sum_{i=2}^M \frac{c_{1i}}{c_{ii}}$$

Aus der Definition der Matrix C folgt $c_{ij} \geq 0$ für alle $i, j \in \{1, \dots, M\}$. Für die Diagonal - Elemente c_{ii} ergibt sich $c_{ii} = 1 + \sum_{k=1}^M p_{ik} c_{ki}$, also $c_{ii} \geq 1$, so daß für die Speicherstrategien g'_2 und g'_3 gilt: $\bar{K}(g'_2) \geq \bar{K}(g'_3)$.

Beispiel:

Für $M = 3$ geben wir ein Zahlenbeispiel an. Sei

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix} \quad \text{Damit ergibt sich } \tilde{P} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}$$

Für die Fundamentalmatrix folgt:

$$C = \begin{pmatrix} \frac{68}{39} & \frac{14}{13} & \frac{8}{13} \\ \frac{40}{39} & \frac{22}{13} & \frac{7}{13} \\ \frac{12}{39} & \frac{12}{13} & \frac{18}{13} \end{pmatrix}$$

Wird den Programmläufen Speicherraum gemäß g_1^1 zugewiesen, so belegen diese konstant 3 Seiten und vor jedem Ausführungsbeginn ist ein Umspeicherprozeß nötig.

Gemäß g_2^1 belegen die Programmläufe konstant 1 Seite; im mittel sind jedoch bei einem Programmlauf 3,4 Umspeicherprozesse auszuführen.

Gemäß g_3^1 belegen die Programmläufe im mittel 2,1 Seiten und es sind 2,1 Umspeicherprozesse auszuführen.

Es sei darauf hingewiesen, daß die Definition der ungetakteten Markoff - Ketten für eine beliebige Zerlegung von $\{1, \dots, N\}$ angegeben werden kann, so daß auch für Speicherstrukturen, die von einer Seiteneinteilung abweichen, Strategien in entsprechender Weise untersucht werden können.

5. Ausblick: Adaptive Betriebsprogramme

In diesem Kapitel wird das Problem der Auswahl einer Strategie für das Betriebsprogramm einer Rechenanlage für den allgemeinen Fall heterogener Umweltanforderungen als Optimierungsaufgabe formuliert. Ist diese universell lösbar, so genügt ein Betriebsprogramm, das eine optimale Strategie realisiert, in jeder Situation allen Anforderungen an die Rechenanlage. I. a. wird eine universell optimale Strategie nicht existieren; für diese Fälle wird vorgeschlagen, Betriebsprogramme zu benutzen, die gemischte Strategien realisieren und damit die Fähigkeit besitzen, sich an verschiedene Umweltbedingungen anzupassen. Ein Beispiel für eine gemischte Strategie, die in diesem Sinne ein adaptives Verhalten ermöglicht, wird aus den Simulationsergebnissen des Kapitels 3 abgeleitet.

Für die allgemeine Formulierung des Problems knüpfen wir an die in Abschnitt 2.1.1. behandelte Aufgabe an. Gegeben sind eine Bedienungsstation (dort stellvertretend für den Rechnerkern) und n Aufträge, die von dieser Bedienungsstation auszuführen sind. Für die Aufträge sind in zwei n -tupeln $\pi = (e_1, \dots, e_n)$ und $\tau = (t_1, \dots, t_n)$ die Ausführungsdauer und die Ankunftsstermine festgelegt. Die Bedienungsstation hat für die Ausführung der Aufträge gewisse Freiheiten; im Rahmen dieser Freiheiten legt eine Strategie S eindeutig fest, in welcher Reihenfolge anstehende Aufträge abzarbeiten sind. Mit der Entscheidung, daß die Bedienungsstation die n Aufträge gemäß einer bestimmten Strategie ausführt, sind alle Parameter, die für die Ausführung der Aufträge oder die Tätigkeit der Bedienungsstation von Interesse sind (also z. B. die Fertigstellungstermine und Wartezeiten der Aufträge bzw. die mittlere Auslastung der Bedienungsstation) festgelegt. Beobachtet man zwei gleiche Bedienungsstationen, die nach verschiedenen Strategien dieselben Auf-

träge ausführen, so möchte man die Tätigkeit der Bedienstungenstationen und damit die Strategien miteinander vergleichen. Hierzu ordnet man jedem Tripel (n, t, S) einen Wert $\psi(n, t, S)$ zu und vergleicht für verschiedene Strategien diese Werte miteinander. In Abhängigkeit von den Zielen, die bei der Auftragsausführung verfolgt werden, wird man die Werte für die Tripel verschieden wählen und entsprechend verschiedene Vergleichsergebnisse erhalten. Diese Vorgehensweise wird im folgenden verallgemeinert.

Sei $\mathcal{R}$ eine reale Rechenanlage fester Struktur und $\mathcal{R}^*$ die ihr zugeordnete virtuelle Rechenanlage mit der Zustandsmenge $Z(\mathcal{R}^*)$. $\mathcal{R}$ erlaube verschiedene Benutzungsmodi, in denen jeweils eine Klasse von Aufträgen behandelt werde. Solche Modi sind z. B. der Betrieb von $\mathcal{R}$ als Teilnehmer-Rechensystem oder der Betrieb von $\mathcal{R}$ als "halboffenes" Rechenzentrum während der Testzeit. Jeder Modus lasse sich charakterisieren durch:

1. eine Verteilungsfunktion $\mathcal{V}(x)$ für die Länge D der Zeitintervalle zwischen der Eingabe zweier aufeinanderfolgender Aufträge;
2. einen Auftrags - Raum $\mathcal{A} = (\Omega, \mathcal{P}(\Omega), P)$.

Arbeitet also die Rechenanlage in einem Modus $(\mathcal{V}(x), \mathcal{A})$ so vergeht zwischen der Ankunft zweier aufeinanderfolgender Aufträge die Zeit D , wobei D der Verteilungsfunktion $\mathcal{V}(x)$ genügt, und mit der Wahrscheinlichkeit $P(\omega)$ besteht ein eingegebener Auftrag in der Aufforderung, den Programmlauf $\omega \in \Omega$ auszuführen.

Insgesamt werde die Benutzung von $\mathcal{R}$ beschrieben durch die Menge Θ von Verteilungsfunktionen $\mathcal{V}(x)$ und die Menge $\mathcal{F}$ von Auftrags - Räumen. Aufträge, die gemäß $(\mathcal{V}, \mathcal{A}) \in \Theta \times \mathcal{F}$ eingegeben werden, könne $\mathcal{R}$ nach Strategien aus einer Menge Σ ausführen.

Durch ein Tripel $(\mathcal{V}, \mathcal{A}, S) \in \Theta \times \mathcal{F} \times \Sigma$ ist die Tätigkeit von $\mathcal{R}$ festgelegt; für alle Aufträge gemäß $\mathcal{A}$ sind damit insbeson-

dere die reale Ausführungsdauer, die Verweilzeit in $\bar{Q}$ die Wartezeit auf Ausführung und ähnliche charakteristische Parameter bestimmt; andererseits sind damit auch charakteristische Parameter von $\bar{Q}$, wie z. B. die mittlere Auslastung des Rechnerkerns und der E/A - Steuerwerke bestimmt.

Ein Vergleich zwischen verschiedenen Strategien setzt eine Ordnungsrelation auf der Menge $\Theta \times \bar{\Phi} \times \Sigma$ voraus. Diese kann nach verschiedenen Bewertungskriterien definiert werden. Wir nehmen an, daß eine Menge Ψ von Abbildungen $\psi: \Theta \times \bar{\Phi} \times \Sigma \rightarrow [0,1]$ derart gegeben ist, daß folgendes gilt: Ist $\psi(\bar{\nu}, \alpha, S)$ das Bild von $(\bar{\nu}, \alpha, S)$ bzgl. ψ und gilt $\psi(\bar{\nu}, \alpha, S) \geq \psi(\bar{\nu}', \alpha', S')$, so ist der Wert der Strategie $S \in \Sigma$ bei der Ausführung von Aufträgen, die gemäß $(\bar{\nu}, \alpha) \in \Theta \times \bar{\Phi}$ eingegeben werden, bzgl. ψ größer ($>$), gleich(=) oder kleiner ($<$), als der Wert der Strategie $S' \in \Sigma$ bei der Ausführung von Aufträgen, die gemäß $(\bar{\nu}', \alpha') \in \Theta \times \bar{\Phi}$ eingegeben werden. (Treten Betriebsmodi $(\bar{\nu}, \alpha)$ und $(\bar{\nu}', \alpha')$ auf, so daß eine Eingabe gemäß $(\bar{\nu}, \alpha')$ oder $(\bar{\nu}', \alpha)$ sinnlos ist, so wird angenommen, daß $\psi(\bar{\nu}, \alpha', S) = \psi(\bar{\nu}', \alpha, S) = 0$ für alle $\psi \in \Psi$ und alle $S \in \Sigma$ festgelegt ist.)

Def. 5.1. Existiert eine Strategie $S' \in \Sigma$ so daß für alle $S \in \Sigma$ $\psi(\bar{\nu}, \alpha, S') \geq \psi(\bar{\nu}, \alpha, S)$ gilt, so heißt S' ψ - $\bar{\nu}$ - α -optimal bzgl. Σ .

In entsprechender Weise sind ψ - $\bar{\nu}$ -optimale, ψ - α -optimale und ψ -optimale Strategien zu definieren.

Def. 5.2. Existiert eine Strategie $S' \in \Sigma$ so daß $\psi(\bar{\nu}, \alpha, S') \geq \psi(\bar{\nu}, \alpha, S)$ gilt für alle $S \in \Sigma$ und alle $(\psi, \bar{\nu}, \alpha) \in \Psi \times \Theta \times \bar{\Phi}$, so heißt S' universell optimale Strategie bzgl. Σ, Ψ, Θ und $\bar{\Phi}$.

Ist ein Betriebsprogramm für die Rechananlage $\mathcal{R}$ zu schreiben, so sind also folgende Aufgaben zu lösen:

1. Bestimmung der für alle Betriebsmodi von $\mathcal{R}$ relevanten Mengen Θ und Φ , sowie der Bewertungskriterien, nach denen die Tätigkeit von $\mathcal{R}$ beurteilt werden soll.
2. Berechnung einer universell optimalen Strategie bzgl. Θ, Φ , Σ und aller Bewertungskriterien, die berücksichtigt werden sollen, wobei Σ die Menge aller von $\mathcal{R}$ bei der Ausführung von Aufträgen gemäß $(\mathcal{N}, \alpha) \in \Theta \times \Phi$ realisierbaren Strategien ist.

Existiert eine universell optimale Strategie $S' \in \Sigma$, so ist diese durch ein Betriebsprogramm zu realisieren; dann arbeitet $\mathcal{R}$ in jedem Betriebsmodus optimal bzgl. aller Bewertungskriterien.

Eine universell optimale Strategie wird i. a. nicht existieren; allein die verschiedenen Bewertungskriterien, nach denen die Tätigkeit von Rechananlagen beurteilt wird, machen dies offenbar. Für diese Fälle sind Strategien von Interesse, die wenigstens für gewisse Untermengen von Θ , Φ und Ψ optimal sind.

Def. 5.3. Existieren eine nichtleere Untermenge $\Gamma \subset \Psi \times \Theta \times \Phi$ und eine Strategie $S' \in \Sigma$ mit:

1. $\psi(\mathcal{N}, \alpha, S') \geq \psi(\mathcal{N}, \alpha, S)$
für alle $(\psi, \mathcal{N}, \alpha) \in \Gamma$ und alle $S \in \Sigma$;
2. ist $\Gamma \neq \Psi \times \Theta \times \Phi$, so existiert für jedes $(\psi, \mathcal{N}, \alpha)$ mit $\{(\psi, \mathcal{N}, \alpha)\} \cap \Gamma \neq \emptyset$
eine Strategie $S'' \in \Sigma$ mit $S'' \neq S'$ und
 $\psi(\mathcal{N}, \alpha, S') < \psi(\mathcal{N}, \alpha, S'')$;
3. es gibt keine Strategie $S'' \in \Sigma$ mit $S'' \neq S'$ und
 $\psi(\mathcal{N}, \alpha, S'') \geq \psi(\mathcal{N}, \alpha, S')$ für alle
 $(\psi, \mathcal{N}, \alpha) \in \Gamma'$ mit $\Gamma' \subset \Psi \times \Theta \times \Phi$ und $\Gamma' \supset_{\text{echt}} \Gamma$;

dann heißt

S' partiell optimal bzgl. $\Psi \times \Theta \times \Phi$
und Σ . Γ heißt Optimalitätsbereich von S' .

Existiert eine universell optimale Strategie bzgl. Ψ, Θ, Φ und Σ , so ist diese offenbar auch partiell optimal und es gibt außer den universell optimalen Strategien keine partiell optimalen.

$\tilde{\Sigma} := \{S_1, \dots, S_n\}$ sei eine Menge von partiell optimalen Strategien, für deren Optimalitätsbereiche Γ_i gelte:

1. $\Gamma_i \neq \Gamma_k$ für alle $i, k \in \{1, \dots, n\}$ mit $i \neq k$;
2. $\bigcup_{i=1}^n \Gamma_i = \Psi \times \Theta \times \Phi$;
3. für jede echte Untermenge $I \subset \{1, \dots, n\}$ folgt: $\bigcup_{i \in I} \Gamma_i \neq \Psi \times \Theta \times \Phi$.

Existiert eine Untermenge $\tilde{\Sigma}$ von Σ mit diesen Eigenschaften, so enthält $\tilde{\Sigma}$ offenbar für jedes Tripel $(\psi, \vartheta, \alpha) \in \Psi \times \Theta \times \Phi$ wenigstens eine Strategie S_i mit $\psi(\vartheta, \alpha, S_i) \geq \psi(\vartheta, \alpha, S)$ für alle $S \in \Sigma$. Dann kann die Aufgabe, ein Betriebsprogramm für die Rechenanlage $\mathcal{R}$ zu schreiben, wie folgt gelöst werden: Man implementiere ein Betriebsprogramm, das die n Strategien von $\tilde{\Sigma}$ realisieren kann. Werden in einem Zeitintervall $[t_1, t_2)$ Aufträge aus einem Auftrags - Raum $\mathcal{A}$ gemäß einer Verteilungsfunktion $\vartheta(x)$ eingegeben und wird die Tätigkeit der Rechenanlage $\mathcal{R}$ während dieser Zeit gemäß ψ bewertet, so führt das Betriebsprogramm diese Aufträge nach einer Strategie S_i aus, deren Optimalitätsbereich $(\psi, \vartheta, \alpha)$ angehört. Nach Voraussetzung existiert wenigstens ein $S_i \in \tilde{\Sigma}$ mit dieser Eigenschaft. Vom Zeitpunkt t_2 an werden Aufträge aus einem Auftrags - Raum $\mathcal{A}'$ gemäß einer Verteilungsfunktion $\vartheta'(x)$ eingegeben und die Bewertung der Tätigkeit von $\mathcal{R}$ erfolge gemäß ψ' . Gilt dann $(\psi', \vartheta', \alpha') \notin \Gamma_i$, so existiert ein $S_k \in \tilde{\Sigma}$ mit $(\psi', \vartheta', \alpha') \in \Gamma_k$. Dann führe das Betriebsprogramm vom Zeitpunkt t_2 an die Aufträge gemäß S_k aus.

Ein Betriebsprogramm, das nach dieser Vorschrift Folgen von Strategien aus der Menge $\tilde{\Sigma}$ realisiert, verhält sich

nach Definition von $\tilde{\Sigma}$ in jedem Zeitpunkt optimal; die Strategienfolge, die sich in Abhängigkeit von den Anforderungen an $\mathcal{R}$ - repräsentiert durch ψ , $\mathcal{V}(x)$ und α - stellt, ersetzt eine universell optimale Strategie bzgl.

Ψ , Θ , Φ und $\tilde{\Sigma}$. Das Betriebsprogramm besitzt die Fähigkeit, sich an die verschiedenen Umwelanforderungen anzupassen und wird in diesem Sinne **a d a p t i v i m k l e i n e n** genannt.

Die Berechnung universell oder partiell optimaler Strategien für eine reale Rechenanlage $\mathcal{R}$ setzt die Kenntnis von Θ , Φ und Ψ für die Strategiemenge $\tilde{\Sigma}$ voraus. Θ und Φ zu bestimmen ist Aufgabe einer eingehenden Betriebsanalyse; mögliche Wege hierzu sind in Klar/Spies (1969) skizziert. Bei bekannten Θ und Φ bleibt die Aufgabe, die Werte der Strategien den verschiedenen Bewertungskriterien entsprechend festzulegen.

Die Ausführungen in den Kapiteln 2, 3 und 4 sind Beiträge zur Lösung dieser Aufgabe. Dabei wurde folgende Methode benutzt: Durch Vernachlässigung gewisser Details der Rechenanlage $\mathcal{R}$ ergibt sich ein Modell $\bar{\mathcal{R}}$ für $\mathcal{R}$. Das Verhalten von $\bar{\mathcal{R}}$ wird so analysiert, daß charakteristische Parameter bzgl. ausgeführter Aufträge und bzgl. der Elemente von $\bar{\mathcal{R}}$ als Funktionen der Auftrags - Eingabe und der Strategien, nach denen die Ausführung gesteuert wird, bekannt sind. Ausgehend von den Ergebnissen dieser Analyse werden die Abbildungen $\psi \in \Psi$ definiert und damit der Wert der Strategien festgelegt. Die Übertragung dieser Analyse - Ergebnisse auf die Rechenanlage $\mathcal{R}$ bedarf der experimentellen Bestätigung. In Anbetracht der vielfältigen Möglichkeiten bei der Definition von Strategien wird man dabei nicht die Gesamtheit der realisierbaren Strategien, sondern Untermengen hiervon untersuchen, so daß sich Optimalitätsaussagen nur für spezielle Mengen von Strategien ergeben.

Beispiel:

Unter der Voraussetzung, daß die in Kapitel 3 definierte Versuchsanordnung ein geeignetes Modell für eine Rechenanlage $\mathcal{R}$ ist, die als Teilnehmer - Rechensystem arbeiten soll, seien:

$$1. \quad \Theta := \{NA \cdot \mathcal{N}(x) \mid NA \in \{1, \dots, 30\}\},$$

wobei $\mathcal{N}(x)$ die in Bild 7 angegebene Verteilungsfunktion für die Denkzeit ist und $NA \cdot \mathcal{N}(x)$ bedeute, daß NA Benutzer aktiv sind, wobei sich jeder einzelne gemäß $\mathcal{N}(x)$ verhalte.

$$2. \quad \Phi := \{\alpha\}$$

wobei α der Auftrags - Raum sei, aus dem in Kapitel 3 die Verteilungsfunktionen $\xi(x)$, $\mathcal{M}(x)$ und $\mathcal{L}(x|m)$ abgeleitet sind.

$$3. \quad \Sigma := \{S_{111}, S_{211}, S_{112}, S_{212}, S_{323}\},$$

wobei S_{ijk} die in Kapitel 3 definierten Strategien sind.

Seien $v(\mathcal{N}', \alpha, S)$ die Verweilzeit eines Auftrages, der gemäß $\mathcal{N}' \in \Theta$ eingegeben und gemäß $S \in \Sigma$ ausgeführt wird, $u(\mathcal{N}', \alpha, S)$ die Anzahl der HP - Umspeicherungen und $c(\mathcal{N}', \alpha, S)$ die Anzahl der belegten Seiten im Primärspeicher unter denselben Voraussetzungen. Dann sei:

$$4. \quad \Psi := \{\psi_1, \psi_2, \psi_3, \psi_4\} \text{ mit}$$

$$\psi_1(\mathcal{N}', \alpha, S) := \frac{1}{1 + \text{Mittelwert} [v(\mathcal{N}', \alpha, S)]}$$

$$\psi_2(\mathcal{N}', \alpha, S) := \frac{1}{1 + \text{Streuung} [v(\mathcal{N}', \alpha, S)]}$$

$$\psi_3(\mathcal{N}', \alpha, S) := \frac{1}{1 + \text{Mittelwert} [u(\mathcal{N}', \alpha, S)]}$$

$$\psi_4(\mathcal{N}', \alpha, S) := \frac{\text{Mittelwert} [c(\mathcal{N}', \alpha, S)]}{M_0}$$

Für $M_0 = 64 \cdot 1024$ Wörter sind die Strategien S_{323} und S_{211} partiell optimal; die Optimalitätsbereiche sind:

$$\begin{aligned} \Gamma(S_{323}) &:= \{\psi_1, \psi_2\} \times \Theta \times \mathbb{I} \cup \{\psi_3\} \times \Theta' \times \mathbb{I} \\ &\quad \text{mit } \Theta' := \{NA \cdot \mathcal{N}(x) \mid NA \in \{1, \dots, 10\}\}; \\ \Gamma(S_{211}) &:= \{\psi_3, \psi_4\} \times \Theta \times \mathbb{I} \end{aligned}$$

Außer der experimentellen Bestätigung der Analyse - Ergebnisse in einer realen Rechenanlage stehen insbesondere Untersuchungen aus, die klären, in welchem Maße sich die Anforderungen an eine Rechenanlage über größere Zeiträume hinweg ändern. Es ist anzunehmen, daß ein fortwährender *Lernprozeß* nötig ist, wenn ein Betriebsprogramm auch diesen Entwicklungen Rechnung tragen soll. In diesen Lernprozeß müßten einerseits ständige Überprüfungen der von den Benutzern an die Rechenanlagen gestellten Anforderungen, andererseits Untersuchungen über den Wert von Strategien einbezogen sein. Ein möglicher Weg dies zu realisieren bestünde in der Integration dieses Lernprozesses in das Betriebsprogramm einer Rechenanlage. Untersuchungen über Betriebsprogramme, die sich in diesem Sinne *adaptiv im großen* verhalten, und insbesondere der Lernstrategien für diese Betriebsprogramme werden in Zukunft durchzuführen sein.

Literatur

- Bauer H. (1968), Wahrscheinlichkeitstheorie und Grundzüge der Maßtheorie, Verlag Walter de Gruyter, Berlin
- Coffman E. G. and Wood R. C. (1966), Interarrival Statistics for Time - Sharing Systems, Communications of the ACM, V 9, N 7, S. 500 - 503
- Conway R. W., Maxwell W. L., Miller L. W. (1967), Theory of Scheduling, Addison - Wesley Publishing Company, Reading
- Doetsch G. (1958), Einführung in Theorie und Anwendung der Laplace - Transformation, Birkhäuser Verlag, Basel
- Estrin G., Hopkins D., Coggan B., Crocker S. (1967), SNUPER - COMPUTER - a computer instrumentation automaton, Spring Joint Computer Conference Proceedings, Spartan Books, Washington
- Evans G. W., Wallace G. F., Sutherland G. L. (1967), Simulation Using Digital Computers, Prentice - Hall, Englewood Cliffs
- Fine G. H., Jackson C. W., McIsaac P. V. (1966), Dynamic Program Behavior under Paging, Proceedings of 21st National Conference ACM, Thompson Book Company, Washington
- Fisz M. (1965), Wahrscheinlichkeitsrechnung und Mathematische Statistik, VEB Deutscher Verlag der Wissenschaften, Berlin
- Händler W. (1967), Digitale Universalrechenautomaten, Taschenbuch der Nachrichtenverarbeitung (Herausgeber K. Steinbuch), Springer Verlag, Berlin
- Händler W. (1968), Teilnehmer - Rechensysteme, Oldenbourg Verlag, München

- Kemeny J. G., Snell J. L. (1960), Finite Markov Chains,
D. Van Nostrand Company, Princeton
- Klar R., Spies P. P. (1969), Adaptive Betriebsprogramme
und ihre Unterstützung durch einen Zählmonitor,
Vortrag im Kolloquium "Betriebsprogrammierung" des
Instituts für Mathematische Maschinen und Datenver-
arbeitung, Erlangen, (erscheint demnächst im Olden-
bourg Verlag, München)
- Kleinrock L. (1966), Sequential Processing Machines (S.P.M.)
Analyzed with a Queuing Theory Model, Journal of the
ACM, V 13, N 2, S. 179 - 193
- Kleinrock L. (1967), Time - Shared Systems: A theoretical
Treatment, Journal of the ACM, V 14, N 2, S. 242-261
- Little J. D. C. (1961), A Proof for the Queuing Formula
 $L = \lambda W$, Operations Research, V 9, N 3, S. 383 - 387
- Martin F. F. (1968), Computer Modeling and Simulation,
John Wiley & Sons, New York
- Naylor T. H., Balintfy J. L., Burdick D. S., Kong Chu (1966),
Computer Simulation Techniques, John Wiley & Sons,
New York
- Nielsen N. R. (1967), The Simulation of Time Sharing
Systems, Communications of the ACM, V 10, N 7, S. 397-412
- Saaty T. L. (1961), Elements of Queueing Theory,
McGraw - Hill Book Company, New York
- Scherr A. L. (1967), An Analysis of Time - Shared Computer
Systems, The M. I. T. Press, Cambridge (Mass.)
- Smith J. L. (1966), An Analysis of Time - Sharing Computer
Systems Using Markov Models, Spring Joint Computer
Conference Proceedings, Spartan Books, Washington

- Smith J. L. (1967), Multiprogramming under a Page on Demand Strategy, Communications of the ACM, V 10, N 10, S. 636 - 646
- Spies P. P. (1967), Teilnehmer - Rechensysteme, Elektronische Rechenanlagen 9, H. 3, S. 117 - 122
- Takacs L. (1962), Introduction to the Theory of Queues, Oxford University Press, New York
- Zurmühl R. (1965), Praktische Mathematik, Springer - Verlag, Berlin

Herrn Prof. Dr. W. Händler möchte ich meinen Dank aussprechen für die Anregung zu dieser Arbeit und ihre Förderung. Allen Kollegen der Arbeitsgruppe "Rechnerstrukturen", insbesondere Herrn Dipl.-Phys. R. Klar, danke ich für wertvolle Hinweise und Diskussionsbemerkungen. Den Mitarbeitern des Rechenzentrums der Universität Erlangen-Nürnberg sei gedankt für die entgegenkommende Betreuung bei der Durchrechnung der Simulationsprogramme.

Erlangen, am 18. 8. 1969

Peter Paul Frits

L e b e n s l a u f

Geboren am 20. September 1939 in Würzburg als Sohn des Maschinisten Peter Spies und seiner Frau Amalie, geb. Feld.

Besuch der kath. Volksschule in Schwalbach/Saar von 1945 bis 1949 und des humanistischen Gymnasiums in Saarlouis von 1949 bis 1958.

Nach dem Abitur Studium der Mathematik an der Universität des Saarlandes in Saarbrücken vom Sommersemester 1958 bis zum Wintersemester 1964/65.

Nach dem Diplom seit Februar 1965 wissenschaftlicher Mitarbeiter von Herrn Prof. Dr. W. Händler am Lehrstuhl für Elektronische Rechenanlagen der Technischen Hochschule Hannover und seit Herbst 1966 am Institut für Mathematische Maschinen und Datenverarbeitung der Universität Erlangen-Nürnberg.

Offsetdruck-Fotodruck

J. Hogn, 8520 Erlangen Hauptstr. 109, (Nähe Martin-Luther-Platz) Tel. 22990
